IBM Spectrum Scale
Version 4 Release 2.0

Advanced Administration Guide

SC23-7032-06

IBM Spectrum Scale
Version 4 Release 2.0

Advanced Administration Guide

SC23-7032-06

Note
Before using this information and the product it supports, read the information in "Notices" on page 425.

This edition applies to version 4 release 2 of the following products, and to all subsequent releases and modifications until otherwise indicated in new editions:

- IBM Spectrum Scale ordered through Passport Advantage® (product number 5725-Q01)
- IBM Spectrum Scale ordered through AAS/eConfig (product number 5641-GPF)
- IBM Spectrum Scale for Linux on z Systems (product number 5725-S28)

Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

IBM welcomes your comments; see the topic "How to send your comments" on page xiv. When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Tables

About this information

This edition applies to IBM Spectrum Scale™ version 4.2 for AIX®, Linux, and Windows.

IBM Spectrum Scale is a file management infrastructure, based on IBM® General Parallel File System (GPFS™) technology, that provides unmatched performance and reliability with scalable access to critical file data.

To find out which version of IBM Spectrum Scale is running on a particular AIX node, enter:

```
lslpp -l gpfs\*
```

To find out which version of IBM Spectrum Scale is running on a particular Linux node, enter:

```
rpm -qa | grep gpfs
```

To find out which version of IBM Spectrum Scale is running on a particular Windows node, open the **Programs and Features** control panel. The IBM Spectrum Scale installed program name includes the version number.

Which IBM Spectrum Scale information unit provides the information you need?

The IBM Spectrum Scale library consists of the information units listed in Table 1.

To use these information units effectively, you must be familiar with IBM Spectrum Scale and the AIX, Linux, or Windows operating system, or all of them, depending on which operating systems are in use at your installation. Where necessary, these information units provide some background information relating to AIX, Linux, or Windows; however, more commonly they refer to the appropriate operating system documentation.

Note: Throughout this documentation, the term "Linux" refers to all supported distributions of Linux, unless otherwise specified.

Table 1. IBM Spectrum Scale library information units

Information unit	**Type of information**	**Intended users**
IBM Spectrum Scale: Administration and Programming Reference	This information unit explains how to do the following: • Use the commands, programming interfaces, and user exits unique to GPFS • Manage clusters, file systems, disks, and quotas • Export a GPFS file system using the Network File System (NFS) protocol	System administrators or programmers of GPFS systems

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Advanced Administration Guide	This information unit explains how to use the following advanced features of GPFS: • Accessing GPFS file systems from other GPFS clusters • Policy-based data management for GPFS • Creating and maintaining snapshots of GPFS file systems • Establishing disaster recovery for your GPFS cluster • Monitoring GPFS I/O performance with the **mmpmon** command • Miscellaneous advanced administration topics	System administrators or programmers seeking to understand and use the advanced features of GPFS
IBM Spectrum Scale: Concepts, Planning, and Installation Guide	This information unit provides information about the following topics: • Introducing GPFS • GPFS architecture • Planning concepts for GPFS • Installing GPFS • Migration, coexistence and compatibility • Applying maintenance • Configuration and tuning • Uninstalling GPFS	System administrators, analysts, installers, planners, and programmers of GPFS clusters who are very experienced with the operating systems on which each GPFS cluster is based

Table 1. IBM Spectrum Scale library information units (continued)

Information unit	Type of information	Intended users
IBM Spectrum Scale: Data Management API Guide	This information unit describes the Data Management Application Programming Interface (DMAPI) for GPFS. This implementation is based on The Open Group's System Management: Data Storage Management (XDSM) API Common Applications Environment (CAE) Specification C429, The Open Group, ISBN 1-85912-190-X specification. The implementation is compliant with the standard. Some optional features are not implemented. The XDSM DMAPI model is intended mainly for a single-node environment. Some of the key concepts, such as sessions, event delivery, and recovery, required enhancements for a multiple-node environment such as GPFS. Use this information if you intend to write application programs to do the following: • Monitor events associated with a GPFS file system or with an individual file • Manage and maintain GPFS file system data	Application programmers who are experienced with GPFS systems and familiar with the terminology and concepts in the XDSM standard
IBM Spectrum Scale: Problem Determination Guide	This information unit contains explanations of GPFS error messages and explains how to handle problems you may encounter with GPFS.	System administrators of GPFS systems who are experienced with the subsystems used to manage disks and who are familiar with the concepts presented in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*

Prerequisite and related information

For updates to this information, see IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html).

For the latest support information, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Conventions used in this information

Table 2 on page xiv describes the typographic conventions used in this information. UNIX file name conventions are used throughout this information.

Note: Users of IBM Spectrum Scale for Windows must be aware that on Windows, UNIX-style file names need to be converted appropriately. For example, the GPFS cluster configuration data is stored in the `/var/mmfs/gen/mmsdrfs` file. On Windows, the UNIX namespace starts under the `%SystemDrive%\cygwin64` directory, so the GPFS cluster configuration data is stored in the `C:\cygwin64\var\mmfs\gen\mmsdrfs` file.

Table 2. Conventions

Convention	Usage
bold	**Bold** words or characters represent system elements that you must use literally, such as commands, flags, values, and selected menu options. Depending on the context, **bold** typeface sometimes represents path names, directories, or file names.
bold underlined	**bold underlined** keywords are defaults. These take effect if you do not specify a different keyword.
`constant width`	Examples and information that the system displays appear in `constant-width` typeface. Depending on the context, `constant-width` typeface sometimes represents path names, directories, or file names.
italic	*Italic* words or characters represent variable values that you must supply. *Italics* are also used for information unit titles, for the first use of a glossary term, and for general emphasis in text.
<*key*>	Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard. For example, **<Enter>** refers to the key on your terminal or workstation that is labeled with the word *Enter*.

\	In command examples, a backslash indicates that the command or coding example continues on the next line. For example:
	`mkcondition -r IBM.FileSystem -e "PercentTotUsed > 90" \`
	`-E "PercentTotUsed < 85" -m p "FileSystem space used"`

{*item*}	Braces enclose a list from which you must choose an item in format and syntax descriptions.
[*item*]	Brackets enclose optional items in format and syntax descriptions.
<**Ctrl-***x*>	The notation <**Ctrl-***x*> indicates a control character sequence. For example, <**Ctrl-c**> means that you hold down the control key while pressing <**c**>.
item...	Ellipses indicate that you can repeat the preceding item one or more times.
|	In *synopsis* statements, vertical lines separate a list of choices. In other words, a vertical line means *Or*. In the left margin of the document, vertical lines indicate technical changes to the information.

How to send your comments

Your feedback is important in helping us to produce accurate, high-quality information. If you have any comments about this information or any other IBM Spectrum Scale documentation, send your comments to the following e-mail address:

mhvrcfs@us.ibm.com

Include the publication title and order number, and, if applicable, the specific location of the information about which you have comments (for example, a page number or a table number).

To contact the IBM Spectrum Scale development organization, send your comments to the following e-mail address:

gpfs@us.ibm.com

Summary of changes

This topic summarizes changes to the IBM Spectrum Scale licensed program and the IBM Spectrum Scale library. Within each information unit in the library, a vertical line (|) to the left of text and illustrations indicates technical changes or additions made to the previous edition of the information.

Summary of changes
for IBM Spectrum Scale version 4 release 2
as updated, November 2015

Changes to this release of the IBM Spectrum Scale licensed program and the IBM Spectrum Scale library include the following:

Cluster Configuration Repository (CCR): Backup and restore
You can backup and restore a cluster that has Cluster Configuration Repository (CCR) enabled. In the **mmsdrbackup** user exit, the type of backup that is created depends on the configuration of the cluster. If the Cluster Configuration Repository (CCR) is enabled, then a CCR backup is created. Otherwise, a mmsdrfs backup is created. In the **mmsdrrestore** command, if the configuration file is a Cluster Configuration Repository (CCR) backup file, then you must specify the `-a` option. All the nodes in the cluster are restored.

Changes in IBM Spectrum Scale for object storage

Object capabilities
Object capabilities describe the object protocol features that are configured in the IBM Spectrum Scale cluster such as unified file and object access, multi-region object deployment, and S3 API emulation. For more information, see the following topics:
- *Object capabilities* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Managing object capabilities* in *IBM Spectrum Scale: Administration and Programming Reference*

Storage policies for object storage
Storage policies enable segmenting of the object storage within a single cluster for various use cases. Currently, OpenStack Swift supports storage polices that allow you to define the replication settings and location of objects in a cluster. IBM Spectrum Scale enhances storage policies to add compression and unified file and object access functions for object storage. For more information, see the following topics:
- *Storage policies for object storage* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Mapping of storage policies to filesets* in *IBM Spectrum Scale: Administration and Programming Reference*
- *Administering storage policies for object storage* in *IBM Spectrum Scale: Administration and Programming Reference*

Multi-region object deployment
The main purpose of the object protocol is to enable the upload and download of object data. When clients have a fast connection to the cluster, the network delay is minimal. However, when client access to object data is over a WAN or a high-latency network, the network can introduce an unacceptable delay and affect quality-of-service metrics. To improve that response time, you can create a replica of the data in a cluster closer to the clients using the active-active multi-region replication support in OpenStack Swift. Multi-region can also be used to distribute the object load over several clusters to reduce contention in the file system. For more information, see the following topics:

- *Overview of multi-region object deployment* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Planning for multi-region object deployment* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Enabling multi-region object deployment initially* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Adding a region in a multi-region object deployment* in *IBM Spectrum Scale: Administration and Programming Reference*
- *Administering a multi-region object deployment environment* in *IBM Spectrum Scale: Administration and Programming Reference*

Unified file and object access

Unified file and object access allows users to access the same data as an object and as a file. Data can be stored and retrieved through IBM Spectrum Scale for object storage or as files from POSIX, NFS, and SMB interfaces. For more information, see the following topics:

- *Unified file and object access overview* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Planning for unified file and object access* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Installing and using unified file and object access* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*
- *Unified file and object access in IBM Spectrum Scale* in *IBM Spectrum Scale: Administration and Programming Reference*

S3 access control lists (ACLs) support

IBM Spectrum Scale for object storage supports S3 access control lists (ACLs) on buckets and objects. For more information, see *Managing OpenStack access control lists using S3 API emulation* in *IBM Spectrum Scale: Administration and Programming Reference*.

Changes in IBM Spectrum Scale for Linux on z Systems™

- Compression support
- AFM-based Async Disaster Recovery (AFM DR) support
- IBM Spectrum Protect™ Backup-Archive and Space Management client support
- Support for all editions:
  - Express®
  - Standard
  - Advanced (without encryption)

For more information about current requirements and limitations of IBM Spectrum Scale for Linux on z Systems, see *Q2.25* of IBM Spectrum Scale FAQ.

Change in AFM-based Async Disaster Recovery (AFM DR)

- Support for IBM Spectrum Scale for Linux on z Systems

File compression

With file compression, you can reclaim some of the storage space occupied by infrequently accessed files. Run the **mmchattr** command or the **mmapplypolicy** command to identify and compress a few files or many files. Run file compression synchronously or defer it. If you defer it, you can run the **mmrestripefile** or **mmrestripefs** to complete the compression. You can decompress files with the same commands used to compress files. When a compressed file is read it is decompressed on the fly and remains compressed on disk. When a compressed file is overwritten, the parts of the file that overlap with the changed data are decompressed on disk synchronously in the granularity of ten data blocks. File compression in this release is designed to be used only for compressing cold data or write-once objects and files. Compressing other types of data can result in performance degradation. File compression uses the zlib data compression library and favors saving space over speed.

GUI servers

The IBM Spectrum Scale system provides a GUI that can be used for managing and monitoring the system. Any server that provides this GUI service is referred to as a GUI server. If you need GUI service in the system, designate at least two nodes as GUI servers in the cluster. A maximum of three nodes can be designated as GUI servers. For more information on installing IBM Spectrum Scale using the GUI, see *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

IBM Spectrum Scale management GUI

The management GUI helps to manage and monitor the IBM Spectrum Scale system. You can perform the following tasks through management GUI:

- Monitoring the performance of the system based on various aspects
- Monitoring system health
- Managing file systems
- Creating filesets and snapshots
- Managing Objects and NFS and SMB data exports
- Creating administrative users and defining roles for the users
- Creating object users and defining roles for them
- Defining default, user, group, and fileset quotas
- Monitoring the capacity details at various levels such as file system, pools, filesets, users, and user groups

Hadoop Support for IBM Spectrum Scale

IBM Spectrum Scale has been extended to work seamlessly in the Hadoop ecosystem and is available through a feature called File Placement Optimizer (FPO). Storing your Hadoop data using FPO allows you to gain advanced functions and the high I/O performance required for many big data operations. FPO provides Hadoop compatibility extensions to replace HDFS in a Hadoop ecosystem, with no changes required to Hadoop applications.

You can deploy a IBM Spectrum Scale using FPO as a file system platform for big data analytics. The topics in this guide covers a variety of Hadoop deployment architectures, including IBM BigInsights®, Platform Symphony®, or with a Hadoop distribution from another vendor to work with IBM Spectrum Scale.

IBM Spectrum Scale offers two kinds of interfaces for Hadoop applications to access File System data. One is IBM Spectrum Scale connector, which aligns with Hadoop Compatible File System architecture and APIs. The other is HDFS protocol, which provides a HDFS compatible interfaces.

For more information, see the following sections in the *IBM Spectrum Scale: Advanced Administration Guide*:

- *Hadoop support for IBM Spectrum Scale*
- *Configuring FPO*
- *Hadoop connector*
- *HDFS protocol*

IBM Spectrum Scale installation GUI

You can use the installation GUI to install the IBM Spectrum Scale system. For more information on how to launch the GUI installer, see the *Installing IBM Spectrum Scale using the graphical user interface (GUI)* section in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Performance Monitoring Tool using the Installation Kit

The usage statement and optional arguments have changed during the installation of the toolkit. The new usage statement with options is as follows:

```
spectrumscale config perfmon [-h] [-l] [-r {on,off}]
```

For more information, see *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Protocols cluster disaster recovery (DR)

You can use the **mmcesdr** command to perform DR setup, failover, failback, backup, and restore actions. Protocols cluster DR uses the capabilities of Active File Management based Async Disaster Recovery (AFM DR) to provide a solution that allows an IBM Spectrum Scale cluster to fail over to another cluster and fail back, and backup and restore the protocol configuration information in cases where a secondary cluster is not available. For more information, see *Protocols cluster disaster recovery* in *IBM Spectrum Scale: Advanced Administration Guide*.

Quality of Service for I/O operations (QoS)

You can use the QoS capability to prevent I/O-intensive, long-running GPFS commands, called *maintenance commands*, from dominating file system performance and significantly delaying normal tasks that also compete for I/O resources. Determine the maximum capacity of your file system in I/O operations per second (IOPS) with the new **mmlsqos** command. With the new **mmchqos** command, assign a smaller share of IOPS to the QoS **maintenance** class, which includes all the maintenance commands. Maintenance command instances that are running at the same time compete for the IOPS allocated to the **maintenance** class, and are not allowed to exceed that limit.

Security mode for new clusters

Starting with IBM Spectrum Scale V4.2, the default security mode for new clusters is AUTHONLY. The **mmcrcluster** command sets the security mode to AUTHONLY when it creates the cluster and automatically generates a public/private key pair for authenticating the cluster. In the AUTHONLY security mode, the sending and receiving nodes authenticate each other with a TLS handshake and then close the TLS connection. Communication continues in the clear. The nodes do not encrypt transmitted data and do not check data integrity.

In IBM Spectrum Scale V4.1 or earlier, the default security mode is EMPTY. If you update a cluster from IBM Spectrum Scale V4.1 to V4.2 or later by running `mmchconfig release=LATEST`, the command checks the security mode. If the mode is EMPTY, the command issues a warning message but does not change the security mode of the cluster.

Snapshots

You can display information about a global snapshot without displaying information about fileset snapshots with the same name. You can display information about a fileset snapshot without displaying information about other snapshots that have the same name but are snapshots of other filesets.

spectrumscale Options

The **spectrumscale** command options for installing GPFS and deploying protocols have changed to remove **config enable** and to add **config perf**. For more information, see *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

New options have been added to **spectrumscale setup** and **spectrumscale deploy** to disable prompting for the encryption/decryption secret. Note that if **spectrumscale setup --storesecret** is used, passwords will not be secure. New properties have been added to **spectrumscale cofig object** for setting password data instead of doing so through **enable object**. For more information, see *IBM Spectrum Scale: Administration and Programming Reference*.

The **spectrumscale** options for managing share ACLs have been added. For more information, see *IBM Spectrum Scale: Administration and Programming Reference*.

ssh and scp wrapper scripts

Starting with IBM Spectrum Scale V4.2, a cluster can be configured to use **ssh** and **scp** wrappers. The wrappers allow GPFS to run on clusters where remote root login through **ssh** is disabled. For more information, see the help topic "Running IBM Spectrum Scale without remote root login" in the *IBM Spectrum Scale: Administration and Programming Reference*.

Documented commands, structures, and subroutines

The following lists the modifications to the documented commands, structures, and subroutines:

New commands

The following commands are new:

- **mmcallhome**
- **mmcesdr**
- **mmchqos**
- **mmlsqos**

New structures

There are no new structures.

New subroutines

There are no new subroutines.

Changed commands

The following commands were changed:

- **mmadddisk**
- **mmaddnode**
- **mmapplypolicy**
- **mmauth**
- **mmbackup**
- **mmces**
- **mmchattr**
- **mmchcluster**
- **mmchconfig**
- **mmchdisk**
- **mmcheckquota**
- **mmchnode**
- **mmcrcluster**
- **mmdefragfs**
- **mmdeldisk**
- **mmdelfileset**
- **mmdelsnapshot**
- **mmdf**
- **mmfileid**
- **mmfsck**
- **mmlsattr**
- **mmlscluster**
- **mmlsconfig**
- **mmlssnapshot**
- **mmnfs**
- **mmobj**
- **mmperfmon**
- **mmprotocoltrace**
- **mmremotefs**
- **mmrestripefile**
- **mmrestripefs**
- **mmrpldisk**
- **mmsdrbackup**

- **mmsdrrestore**
- **mmsmb**
- **mmuserauth**
- **spectrumscale**

Changed structures
There are no changed structures.

Changed subroutines
There are no changed subroutines.

Deleted commands
There are no deleted commands.

Deleted structures
There are no deleted structures.

Deleted subroutines
There are no deleted subroutines.

Messages
The following lists the new, changed, and deleted messages:

New messages
6027-2354, 6027-2355, 6027-2356, 6027-2357, 6027-2358, 6027-2359, 6027-2360, 6027-2361, 6027-2362, 6027-3913, 6027-3914, 6027-3107, 6027-4016, 6027-3317, 6027-3318, 6027-3319, 6027-3320, 6027-3405, 6027-3406, 6027-3582, 6027-3583, 6027-3584, 6027-3585, 6027-3586, 6027-3587, 6027-3588, 6027-3589, 6027-3590, 6027-3591, 6027-3592, 6027-3593

Changed messages
6027-2299, 6027-887, 6027-888

Deleted messages
None.

Chapter 1. Accessing a remote GPFS file system

GPFS allows users shared access to files in either the cluster where the file system was created, or other GPFS clusters. File system access by the cluster where the file system was created is implicit.

The ability to access and mount GPFS file systems owned by other clusters in a network of sufficient bandwidth is accomplished using the **mmauth**, **mmremotecluster** and **mmremotefs** commands. Each site in the network is managed as a separate cluster, while allowing shared file system access.

The cluster owning the file system is responsible for administering the file system and granting access to other clusters on a per cluster basis. After access to a particular file system has been granted to nodes in another GPFS cluster, the nodes can mount the file system and perform data operations as if the file system were locally owned.

Each node in the GPFS cluster requiring access to another cluster's file system must be able to open a TCP/IP connection to every node in the other cluster.

Nodes in two separate remote clusters mounting the same file system are not required to be able to open a TCP/IP connection to each other. For example, if a node in `clusterA` mounts a file system from `clusterB`, and a node in `clusterC` desires to mount the same file system, nodes in `clusterA` and `clusterC` do not have to communicate with each other.

Each node in the GPFS cluster requiring file system access must have one of the following:
- A virtual connection to the file system data through an NSD server (refer to Figure 1 on page 2).
- A physical connection to the disks containing file system data (refer to Figure 2 on page 2).

In this example, network connectivity is required from the nodes in `clusterB` to all the nodes in `clusterA` even if the nodes in `clusterB` can access the disks in `clusterA` directly.

Note: Even when remote nodes have direct connectivity to the SAN, they will still use a connection through an NSD server for any NSDs that have been configured with Persistent Reserve (PR). If you want the remote nodes to access the disks through their direct connection to the SAN, you must ensure that PR is not enabled on the NSDs. See the topic about enabling and disabling Persistent Reserve in the *IBM Spectrum Scale: Administration and Programming Reference*.

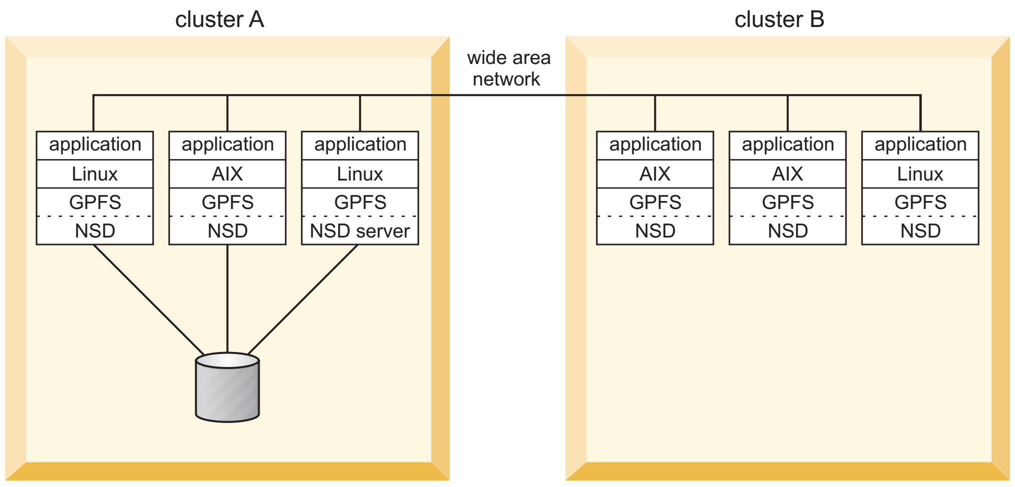


Figure 1. Remote mount of a file system using NSD server access

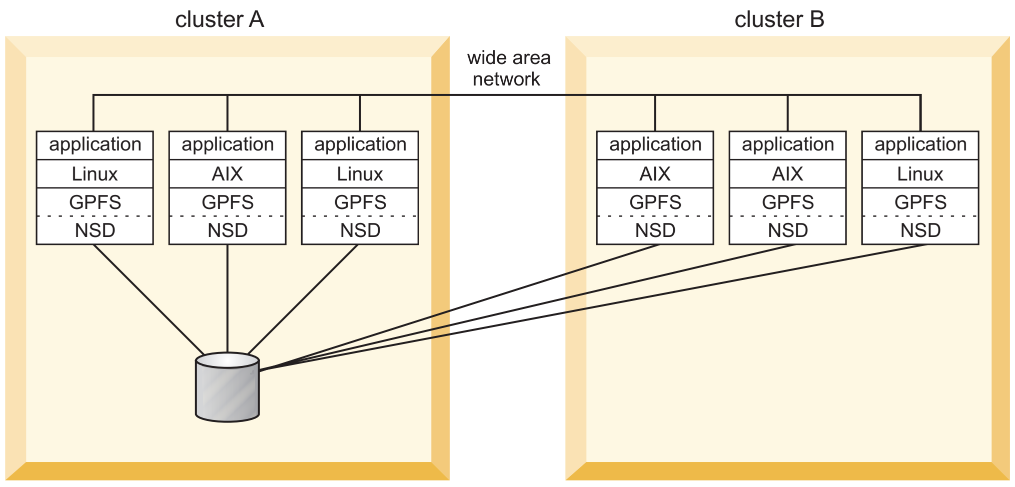


Figure 2. Remote mount of a file system using SAN-attached disks

Figure 3 on page 3 illustrates a multi-cluster configuration with multiple NSD servers. In this configuration:

- The two nodes in Cluster 1 are defined as the NSD servers (you can have up to eight NSD server nodes).
- All three clusters are connected with Gigabit Ethernet.
- Cluster 1 shares an InfiniBand switch network with Cluster 2 and an InfiniBand switch network with Cluster 3.

Note: Protocol support is not available in a multi-cluster configuration.

In order to take advantage of the fast networks and to use the nodes in Cluster 1 as NSD servers for Cluster 2 and Cluster 3, you must configure a subnet for each of the supported clusters. For example issuing the command:

- **mmchconfig subnets="**<*IB_Network_1*> <*IB_Network_1*>**/Cluster1"** in Cluster 2 allows nodes N_2 through N_x to use N_1 as an NSD server with InfiniBand Network 1 providing the path to the data.
- **mmchconfig subnets="**<*IB_Network_2*> <*IB_Network_2*>**/Cluster1"** in Cluster 3 allows nodes N_{2+x} through N_{y+x} to use N_{1+x} as an NSD server with InfiniBand Network 2 providing the path to the data.

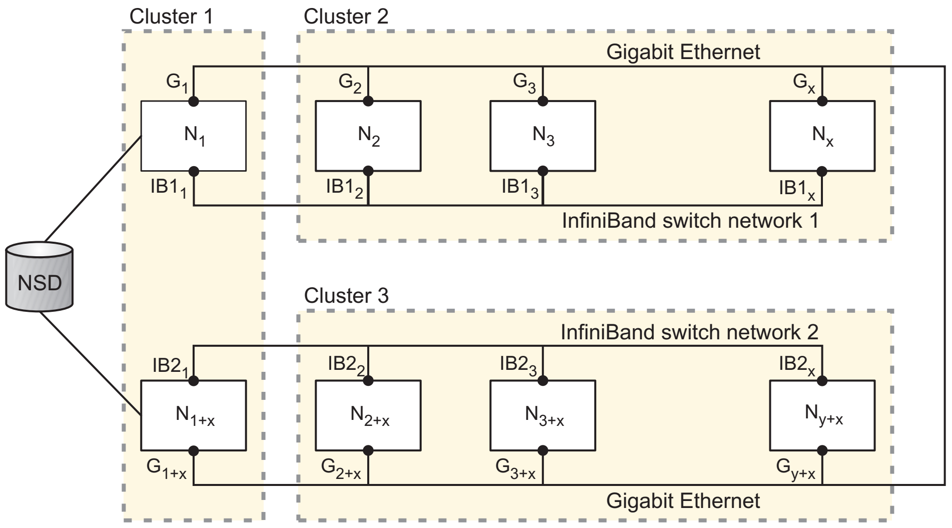


Figure 3. Multi-cluster configuration with multiple NSD servers

When you implement file access from other clusters, consider these topics:

- "Remote user access to a GPFS file system"
- "Mounting a remote GPFS file system" on page 4
- "Managing remote access to a GPFS file system" on page 6
- "Using remote access with public and private IP addresses" on page 7
- "Using multiple security levels for remote access" on page 9
- "Changing security keys with remote access" on page 10
- "Important information about remote access" on page 11

Remote user access to a GPFS file system

In a cluster environment that has a single user identity namespace, all nodes have user accounts set up in a uniform manner. This is usually accomplished by having equivalent **/etc/passwd** and **/etc/group** files on all nodes in the cluster.

For consistency of ownership and access control, a uniform user identity namespace is preferred. For example, if user Jane Doe has an account on nodeA with the user name **janedoe** and user ID **1001** and group ID **500**, on all other nodes in the same cluster Jane Doe will have an account with the same user and group IDs. GPFS relies on this behavior to perform file ownership and access control tasks.

If a GPFS file system is being accessed from a node belonging to another GPFS cluster, the assumption about the uniform user account infrastructure might no longer be valid. Since different clusters can be administered by different organizations, it is possible for each of the clusters to have a unique set of user accounts. This presents the problem of how to permit users to access files in a file system owned and served by another GPFS cluster. In order to have such access, the user must be somehow known to the other cluster. This is usually accomplished by creating a user account in the other cluster, and giving this account the same set of user and group IDs that the account has in the cluster where the file system was created.

To continue with this example, Jane Doe would need an account with user ID **1001** and group ID **500** created in every other GPFS cluster from which remote GPFS file system access is desired. This approach is commonly used for access control in other network file systems, (for example, NFS or AFS™), but might pose problems in some situations.

For example, a problem arises if Jane Doe already has an account in some other cluster, but the user ID associated with this account is not **1001**, and another user in the other cluster has user ID **1001**. It would require a considerable effort on the part of system administrator to ensure that Jane Doe's account has the same set of IDs on all clusters. It is more desirable to be able to use the existing accounts without having to make changes. GPFS helps to solve this problem by optionally performing user ID and group ID remapping internally, using user-supplied helper applications. For a detailed description of the GPFS user ID remapping convention, see the IBM white paper entitled *UID Mapping for GPFS in a Multi-cluster Environment* in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSFKCN/com.ibm.cluster.gpfs.doc/gpfs_uid/uid_gpfs.html).

Access from a remote cluster by a root user presents a special case. It is often desirable to disallow root access from a remote cluster while allowing regular user access. Such a restriction is commonly known as root squash. A root squash option is available when making a file system available for mounting by other clusters using the **mmauth** command. This option is similar to the NFS root squash option. When enabled, it causes GPFS to squash superuser authority on accesses to the affected file system on nodes in remote clusters.

This is accomplished by remapping the credentials: user id (UID) and group id (GID) of the root user, to a UID and GID specified by the system administrator on the home cluster, for example, the UID and GID of the user nobody. In effect, root squashing makes the root user on remote nodes access the file system as a non-privileged user.

Although enabling root squash is similar to setting up UID remapping, there are two important differences:

1. While enabling UID remapping on remote nodes is an option available to the remote system administrator, root squashing need only be enabled on the local cluster, and it will be enforced on remote nodes. Regular UID remapping is a user convenience feature, while root squashing is a security feature.
2. While UID remapping requires having an external infrastructure for mapping between local names and globally unique names, no such infrastructure is necessary for enabling root squashing.

When both UID remapping and root squashing are enabled, root squashing overrides the normal UID remapping mechanism for the root user.

Mounting a remote GPFS file system

This is an example of how to mount a file system owned and served by another IBM Spectrum Scale cluster. The gpfs.gskit package must be installed on all nodes in the involved clusters before using these instructions.

The procedure to set up remote file system access involves the generation and exchange of authorization keys between the two clusters. In addition, the administrator of the GPFS cluster that owns the file system needs to authorize the remote clusters that are to access it, while the administrator of the GPFS cluster that seeks access to a remote file system needs to define to GPFS the remote cluster and file system whose access is desired.

In this example, **cluster1** is the name of the cluster that owns and serves the file system to be mounted, and **cluster2** is the name of the cluster that desires to access the file system.

Note: The following example uses AUTHONLY as the authorization setting. When you specify AUTHONLY for authentication, GPFS checks network connection authorization. However, data sent over the connection is not protected.

1. On **cluster1**, the system administrator issues the **mmauth** command to generate a public/private key pair. The key pair is placed in **/var/mmfs/ssl**:

   ```
   mmauth genkey new
   ```

2. On **cluster1**, the system administrator enables authorization by issuing:

   ```
   mmauth update . -l AUTHONLY
   ```
3. The system administrator of **cluster1** now gives the file **/var/mmfs/ssl/id_rsa.pub** to the system administrator of **cluster2**, who desires to access the **cluster1** file systems. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.
4. On **cluster2**, the system administrator issues the **mmauth** command to generate a public/private key pair. The key pair is placed in **/var/mmfs/ssl**:

   ```
   mmauth genkey new
   ```
5. On **cluster2**, the system administrator enables authorization by issuing:

   ```
   mmauth update . -l AUTHONLY
   ```
6. The system administrator of **cluster2** gives file **/var/mmfs/ssl/id_rsa.pub** to the system administrator of **cluster1**. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.
7. On **cluster1**, the system administrator issues the **mmauth add** command to authorize **cluster2** to mount file systems owned by **cluster1** utilizing the key file received from the administrator of **cluster2**:

   ```
   mmauth add cluster2 -k cluster2_id_rsa.pub
   ```

   where:

   cluster2
   Is the real name of **cluster2** as given by the **mmlscluster** command on a node in **cluster2**.

   cluster2_id_rsa.pub
   Is the name of the file obtained from the administrator of **cluster2** in Step 6.
8. On **cluster1**, the system administrator issues the **mmauth grant** command to authorize **cluster2** to mount specific file systems owned by **cluster1**:

   ```
   mmauth grant cluster2 -f /dev/gpfs
   ```
9. On **cluster2**, the system administrator now must define the cluster name, contact nodes and public key for **cluster1**:

   ```
   mmremotecluster add cluster1 -n node1,node2,node3 -k cluster1_id_rsa.pub
   ```

   where:

   cluster1
   Is the real name of **cluster1** as given by the **mmlscluster** command on a node in **cluster1**.

   *node1***,** *node2***, and** *node3*
   Are nodes in **cluster1**. The hostname or IP address that you specify must refer to the communications adapter that is used by GPFS as given by the **mmlscluster** command on a node in **cluster1**.

   cluster1_id_rsa.pub
   Is the name of the file obtained from the administrator of **cluster1** in Step 3.

   This permits the cluster desiring to mount the file system a means to locate the serving cluster and ultimately mount its file systems.
10. On **cluster2**, the system administrator issues one or more **mmremotefs** commands to identify the file systems in **cluster1** that are to be accessed by nodes in **cluster2**:

    ```
    mmremotefs add /dev/mygpfs -f /dev/gpfs -C cluster1 -T /mygpfs
    ```

    where:

    /dev/mygpfs
    Is the device name under which the file system will be known in **cluster2**.

/dev/gpfs
: Is the actual device name for the file system in **cluster1**.

cluster1
: Is the real name of **cluster1** as given by the **mmlscluster** command on a node in **cluster1**.

/mygpfs
: Is the local mount point in **cluster2**.

11. On **cluster2**, the command:

    ```
    mmmount /dev/mygpfs
    ```

    then mounts the file system.

Table 3 summarizes the commands that the administrators of the two clusters need to issue so that the nodes in **cluster2** can mount the remote file system **fs1**, owned by **cluster1**, assigning **rfs1** as the local name with a mount point of **/rfs1**.

Table 3. Summary of commands to set up cross-cluster file system access.

cluster1	**cluster2**
mmauth genkey new **mmauth update . -l AUTHONLY**	**mmauth genkey new** **mmauth update . -l AUTHONLY**
Exchange public keys (file **/var/mmfs/ssl/id_rsa.pub**)	
mmauth add *cluster2 ...* **mmauth grant** *cluster2* **-f** fs1 ...	**mmremotecluster add** *cluster1 ...* **mmremotefs add** rfs1 **-f** fs1 **-C** cluster1 **-T** /rfs1

Managing remote access to a GPFS file system

This is an example of how to manage remote access to GPFS file systems.

To see a list of all clusters authorized to mount file systems owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth show
```

To authorize a third cluster, say **cluster3**, to access file systems owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth add cluster3 -k cluster3_id_rsa.pub
mmauth grant cluster3 -f /dev/gpfs1
```

To subsequently revoke **cluster3** authorization to access a specific file system **gpfs1** owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth deny cluster3 -f /dev/gpfs1
```

To completely revoke **cluster3** authorization to access file systems owned by **cluster1**, the administrator of **cluster1** issues this command:

```
mmauth delete cluster3
```

Using remote access with public and private IP addresses

GPFS permits the use of both public and private IP address. Private IP addresses are typically used to communicate on private networks.

Private IP addresses are on one of these subnets:

- 10.0.0.0
- 172.16.0.0
- 192.168.0.0

See RFC 1597 - Address Allocation for Private Internets (www.ip-doc.com/rfc/rfc1597) for more information.

Use the **mmchconfig** command, **subnets** attribute, to specify the private IP addresses to be accessed by GPFS.

Figure 4 on page 9 describes an AIX cluster named **CL1** with nodes named **CL1N1**, **CL1N2**, and so forth, a Linux cluster named **CL2** with nodes named **CL2N1**, **CL2N2**, and another Linux cluster named **CL3** with a node named **CL3N1**. Both Linux clusters have public Ethernet connectivity, and a Gigabit Ethernet configured with private IP addresses (10.200.0.1 through 10.200.0.24), not connected to the public Ethernet. The InfiniBand Switch on the AIX cluster **CL1** is configured using public IP addresses on the 7.2.24/13 subnet and is accessible from the outside.

With the use of both public and private IP addresses for some of the nodes, the setup works as follows:

1. All clusters must be created using host names or IP addresses that correspond to the public network.
2. Using the **mmchconfig** command for the **CL1** cluster, add the attribute: **subnets=7.2.24.0**.

   This allows all **CL1** nodes to communicate using the InfiniBand Switch. Remote mounts between **CL2** and **CL1** will use the public Ethernet for TCP/IP communication, since the **CL2** nodes are not on the 7.2.24.0 subnet.

   GPFS assumes subnet specifications for private networks are independent between clusters (private networks are assumed not physically connected between clusters). The remaining steps show how to indicate that a private network is shared between clusters.
3. Using the **mmchconfig** command for the **CL2** cluster, add the **subnets='10.200.0.0/CL2.kgn.ibm.com;CL3.kgn.ibm.com'** attribute. Alternatively, regular expressions are allowed here, such as **subnets='10.200.0.0/CL[23].kgn.ibm.com'**. See note 2 on page 8 for the syntax allowed for the regular expressions.

   This attribute indicates that the private 10.200.0.0 network extends to all nodes in clusters **CL2** or **CL3**. This way, any two nodes in the **CL2** and **CL3** clusters can communicate through the Gigabit Ethernet.

   This setting allows all **CL2** nodes to communicate over their Gigabit Ethernet. Matching **CL3.kgn.ibm.com** with the cluster list for 10.200.0.0 allows remote mounts between clusters **CL2** and **CL3** to communicate over their Gigabit Ethernet.
4. Using the **mmchconfig** command for the **CL3** cluster, add the **subnets='10.200.0.0/CL3.kgn.ibm.com;CL2.kgn.ibm.com'** attribute, alternatively **subnets='10.200.0.0/CL[32].kgn.ibm.com'**.

   This attribute indicates that the private 10.200.0.0 network extends to all nodes in clusters **CL2** or **CL3**. This way, any two nodes in the **CL2** and **CL3** clusters can communicate through the Gigabit Ethernet.

   Matching of **CL3.kgn.ibm.com** with the cluster list for 10.200.0.0 allows all **CL3** nodes to communicate over their Gigabit Ethernet, and matching **CL2.kgn.ibm.com** with that list allows remote mounts between clusters **CL3** and **CL2** to communicate over their Gigabit Ethernet.

Use the **subnets** attribute of the **mmchconfig** command when you wish the GPFS cluster to leverage additional, higher performance network connections that are available to the nodes in the cluster, or between clusters.

Notes:

1. Use of the **subnets** attribute does not ensure a highly available system. If the GPFS daemon is using the IP address specified by the **subnets** attribute, and that interface goes down, GPFS does not switch to the other network. You can use **mmdiag --network** to verify that the subnet is in fact being used.
2. Each subnet can be listed at most once in each cluster. For example, specifying:

   ```
   subnets='10.200.0.0/CL2.kgn.ibm.com 10.200.0.0/CL3.kgn.ibm.com'
   ```

   where the 10.200.0.0 subnet is listed twice, is not allowed. Therefore, subnets that span multiple clusters have to be assigned a cluster name pattern or a semicolon-separated cluster name list. It is possible to combine these, for example, items in semicolon-separated cluster lists can be plain names or regular expressions, as in the following:

   ```
   subnets='1.0.0.1/CL[23].kgn.ibm.com;OC.xyz.ibm.com'
   ```

   The following shows examples of patterns that are accepted:

   ```
   [af3] matches letters 'a' and 'f', and number 3
   [0-7] matches numbers 0, 1, ... 7
   [a-p0-7] matches letter a, b, ... p and numbers from 0 to 7 inclusive
   * matches any sequence of characters
   ? matches any (one) character
   ```

   If the **subnets** attribute lists multiple subnets, and there are multiple subnets in common between the local cluster and a given remote cluster, then the first subnet in common in the list is used for communications between the local and remote clusters. As an example, suppose that the **subnets** attribute is set as follows, on cluster **CL2.kgn.ibm.com**:

   ```
   subnets='10.200.0.0/CL[23].kgn.ibm.com 10.201.0.0/CL[23].kgn.ibm.com'
   ```

   If node **CL2N1** on cluster **CL2.kgn.ibm.com** has network interfaces with IP addresses 10.200.0.1 and 10.201.0.1, and node **CLN31** on cluster **CL3.kgn.ibm.com** has network interfaces with IP addresses 10.200.0.5 and 10.201.0.5, then the communication between these two nodes will flow over the 10.200.0.0 subnet, with **CL2N1** using the interface with IP address 10.200.0.1, and **CLN31** using the interface with IP address 10.200.0.5.

   Specifying a cluster name or a cluster name pattern for each subnet is only needed when a private network is shared across clusters. If the use of a private network is confined within the local cluster, then no cluster name is required in the subnet specification.

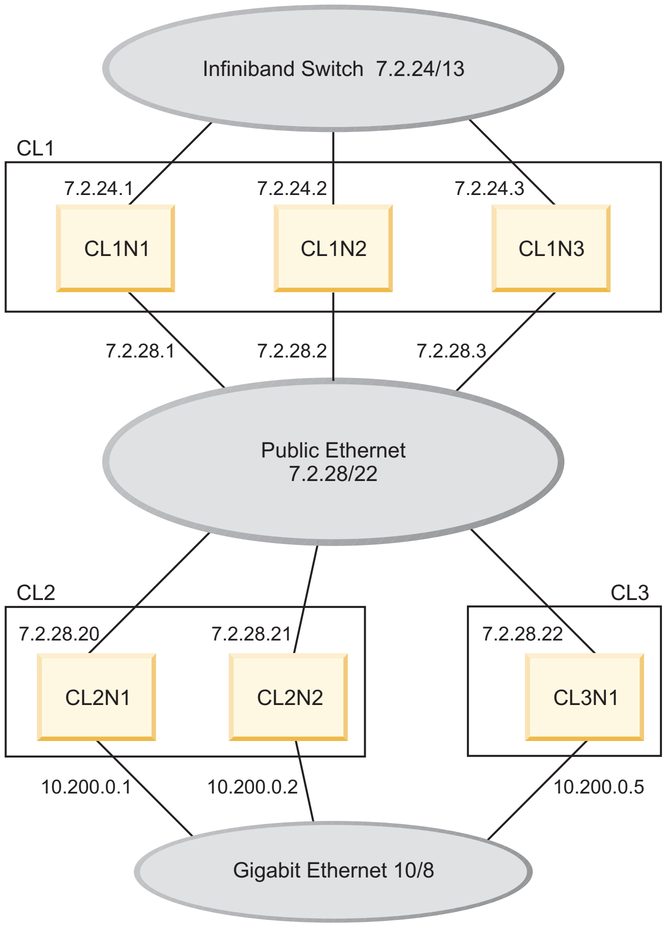


Figure 4. Use of public and private IP addresses in three GPFS clusters

Using multiple security levels for remote access

A cluster that owns a file system whose access is to be permitted from other clusters, can designate a different security level for each connecting cluster.

When multiple security levels are specified, the following rule applies: each connection uses the security level of the connecting node, unless that security level is **AUTHONLY**. In this case, the security level of the node accepting the connection is used instead. This means that a connection will use **AUTHONLY** if and only if both nodes exist in clusters that are required to use security method **AUTHONLY**.

To specify a different security level for different clusters requesting access to a given cluster, use the **mmauth -l** *cipherList* command. Several examples follow to illustrate:

1. In this example, **cluster1** and **cluster2** are located on the same trusted network, and **cluster3** is connected to both of them with an untrusted network. The system administrator chooses these security levels:
   - A *cipherList* of **AUTHONLY** for connections between **cluster1** and **cluster2**
   - A *cipherList* of **AES128-SHA** for connections between **cluster1** and **cluster3**
   - A *cipherList* of **AES128-SHA** for connections between **cluster2** and **cluster3**

   The administrator of **cluster1** issues these commands:
   ```
   mmauth add cluster2 -k keyFile -l AUTHONLY
   mmauth add cluster3 -k keyFile -l AES128-SHA
   ```

2. In this example, **cluster2** is accessing file systems owned by **cluster1** using a *cipherList* of **AUTHONLY**, but the administrator of **cluster1** has decided to require a more secure *cipherList*. The administrator of **cluster1** issues this command:

   ```
   mmauth update cluster2 -l AES128-SHA
   ```

   Existing connections will be upgraded from **AUTHONLY** to **AES128-SHA**.

Changing security keys with remote access

When working with GPFS file systems accessed by other GPFS clusters, it might be necessary to generate a new public/private access key. This can be done without disturbing existing connections, provided the following procedure is followed.

To accomplish this, the cluster that owns and serves the file system is made to temporarily have two access keys (referred to as the 'old key' and the 'new key'), which are both valid at the same time. The clusters currently accessing the file system can then change from the old key to the new key without interruption of file system access.

In this example, **cluster1** is the name of the cluster that owns and serves a file system, and **cluster2** is the name of the cluster that has already obtained access to this file system, and is currently using it. Here, the system administrator of **cluster1** changes the access key without severing the connection obtained by **cluster2**.

1. On **cluster1**, the system administrator issues the **mmauth genkey new** command to generate a new public/private access key pair. The key pair is placed in **/var/mmfs/ssl**:

   ```
   mmauth genkey new
   ```

   After this command is issued, **cluster1** will have two keys (referred to as the 'old key' and the 'new key') that can both be used to access **cluster1** file systems.
2. The system administrator of **cluster1** now gives the file **/var/mmfs/ssl/id_rsa.pub** (that contains the new key) to the system administrator of **cluster2**, who desires to continue to access the **cluster1** file systems. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.
3. On **cluster2**, the system administrator issues the **mmremotecluster update** command to make the new key known to his system:

   ```
   mmremotecluster update cluster1 -k cluster1_id_rsa.pub
   ```

   where:

   cluster1
   Is the real name of **cluster1** as given by the **mmlscluster** command on a node in **cluster1**.

   cluster1_id_rsa.pub
   Is the name of the file obtained from the administrator of **cluster1** in Step 2.

   This permits the cluster desiring to mount the file system to continue mounting file systems owned by **cluster1**.
4. On **cluster1**, the system administrator verifies that all clusters desiring to access **cluster1** file systems have received the new key and activated it using the **mmremotecluster update** command.
5. On **cluster1**, the system administrator issues the **mmauth genkey commit** command to commit the new key as the only valid access key. The old key will no longer be accepted once this command completes successfully:

   ```
   mmauth genkey commit
   ```

   Once the new public key has been committed, the old public key will no longer be accepted. As a result, any remote cluster administrator who has not been given the new key (see the preceding Step 2) and run **mmremotecluster update** (see the preceding Step 3) will no longer be able to mount file systems owned by **cluster1**.

Similarly, the administrator of **cluster2** might decide to change the access key for **cluster2**:

1. On **cluster2**, the system administrator issues the **mmauth genkey new** command to generate a new public/private access key pair. The key pair is placed in **/var/mmfs/ssl**:

   ```
   mmauth genkey new
   ```

   After this command is issued, **cluster2** will have two keys (referred to as the 'old key' and the 'new key') that can both be used when a connection is established to any of the nodes in **cluster2**.
2. The system administrator of **cluster2** now gives the file **/var/mmfs/ssl/id_rsa.pub** (that contains the new key) to the system administrator of **cluster1**, the owner of the file systems. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.
3. On **cluster1**, the system administrator issues the **mmauth update** command to make the new key known to his system:

   ```
   mmauth update cluster2 -k cluster2_id_rsa.pub
   ```

   where:

   cluster2
      Is the real name of **cluster2** as given by the **mmlscluster** command on a node in **cluster2**.

   cluster2_id_rsa.pub
      Is the name of the file obtained from the administrator of **cluster2** in Step 2.

   This permits the cluster desiring to mount the file system to continue mounting file systems owned by **cluster1**.
4. The system administrator of **cluster2** verifies that the administrator of **cluster1** has received the new key and activated it using the **mmauth update** command.
5. On **cluster2**, the system administrator issues the **mmauth genkey commit** command to commit the new key as the only valid access key. The old key will no longer be accepted once this command completes successfully:

   ```
   mmauth genkey commit
   ```

NIST compliance

The **nistCompliance** configuration variable allows the system administrator to restrict the set of available algorithms and key lengths to a subset of those approved by NIST.

The **nistCompliance** variable applies to security transport (tscomm security, key retrieval) only, not to encryption, which always uses NIST-compliant mechanisms.

For the valid values for **nistCompliance**, see the *mmchconfig command* topic in *IBM Spectrum Scale: Administration and Programming Reference*.

Important information about remote access

There is some additional information about this topic that you should take into consideration.

When working with GPFS file systems accessed by nodes that belong to other GPFS clusters, consider the following points:

1. A file system is administered only by the cluster where the file system was created. Other clusters may be allowed to mount the file system, but their administrators cannot add or delete disks, change characteristics of the file system, enable or disable quotas, run the **mmfsck** command, and so forth. The only commands that other clusters can issue are list type commands, such as: **mmlsfs**, **mmlsdisk**, **mmlsmount**, and **mmdf**.

2. Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters, like there is between the nodes within a cluster.

   This means that if the administrator of **cluster1** (the owner of file system **gpfs1**) decides to delete it or rename it, the information for **gpfs1** in **cluster2** becomes obsolete, and an attempt to mount **gpfs1** from **cluster2** will fail. It is assumed that when such changes take place, the two administrators will inform each other. The administrator of **cluster2** can then use the **update** or **delete** options of the **mmremotefs** command to make the appropriate changes.
3. If the names of the contact nodes change, the name of the cluster changes, or the public key file changes, use the **update** option of the **mmremotecluster** command to reflect the changes.
4. Use the **show** option of the **mmremotecluster** and **mmremotefs** commands to display the current information about remote clusters and file systems.
5. If the cluster that owns a file system has a **maxblocksize** configuration parameter that is different from the **maxblocksize** configuration parameter of the cluster that desires to mount a file system, a mismatch may occur and file system mount requests may fail with messages to this effect. Check your **maxblocksize** configuration parameters on both clusters using the **mmlsconfig** command. Correct any discrepancies with the **mmchconfig** command.

Chapter 2. Information Lifecycle Management for IBM Spectrum Scale

IBM Spectrum Scale can help you achieve Information Lifecycle Management (ILM) efficiencies through powerful policy-driven automated tiered storage management. With the ILM toolkit, you can manage sets of files and pools of storage, and you can automate the management of file data.

Using these tools, GPFS can automatically determine where to physically store your data regardless of its placement in the logical directory structure. Storage pools, filesets and user-defined policies provide the ability to match the cost of your storage resources to the value of your data.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

GPFS policy-based ILM tools allow you to:

- Create *storage pools* to provide a way to partition a file system's storage into collections of disks or a redundant array of independent disks (RAIDs) with similar properties that are managed together as a group. GPFS has three types of storage pools:
  - A required **system** storage pool that you create and manage through GPFS
  - Optional user storage pools that you create and manage through GPFS
  - Optional external storage pools that you define with GPFS policy rules and manage through an external application such as Tivoli® Storage Manager
- Create *filesets* to provide a way to partition the file system namespace to allow administrative operations at a finer granularity than that of the entire file system. See “Filesets” on page 55.
- Create *policy rules* based on data attributes to determine initial file data placement and manage file data placement throughout the life of the file. See “Policies for automating file management” on page 20.

To work with ILM in the GUI, click **Files** > **Information Lifecycle**.

Use the following information to create and manage information lifecycle management policies in IBM Spectrum Scale:

Storage pools

Physically, a *storage pool* is a collection of disks or RAID arrays. Storage pools also allow you to group multiple storage systems within a file system.

Using storage pools, you can create tiers of storage by grouping storage devices based on performance, locality, or reliability characteristics. For example, one pool could be an enterprise class storage system that hosts high-performance Fibre Channel disks and another pool might consist of numerous disk controllers that host a large set of economical SATA disks.

There are two types of storage pools in GPFS, internal storage pools and external storage pools. Internal storage pools are managed within GPFS. External storage pools are managed by an external application such as Tivoli Storage Manager. For external storage pools, GPFS provides tools that allow you to define an interface that your external storage manager uses to access your data. GPFS does not manage the data placed in external storage pools. Instead, GPFS manages the movement of data to and from external storage pools. Storage pools allow you to perform complex operations such as moving, mirroring, or deleting files across multiple storage devices, providing storage virtualization and a single management context.

Internal GPFS storage pools are meant for managing online storage resources. External storage pools are intended for use as near-line storage and for archival and backup operations. However, both types of storage pools provide you with a method to partition file system storage for considerations such as:

- Improved price-performance by matching the cost of storage to the value of the data
- Improved performance by:
  - Reducing the contention for premium storage
  - Reducing the impact of slower devices
  - Allowing you to retrieve archived data when needed
- Improved reliability by providing for:
  - Replication based on need
  - Better failure containment
  - Creation of new storage pools as needed

For additional information, refer to:

- "Internal storage pools"
- "External storage pools" on page 19

Internal storage pools

Internal GPFS storage pools are controlled by GPFS policies and commands. There are two types of internal GPFS storage pools, the required system storage pool and up to seven optional user storage pools. The system storage pool contains metadata for each file and may also contain user data. User storage pools can only contain user data.

The internal GPFS storage pool to which a disk belongs is specified as an attribute of the disk in the GPFS cluster. You specify the disk attributes as a field in each disk descriptor when you create the file system or when adding disks to an existing file system. GPFS allows a maximum of eight internal storage pools per file system. One of these storage pools is the required **system** storage pool. The other seven internal storage pools are optional user storage pools.

GPFS assigns file data to internal storage pools under these circumstances:

- When the file is initially created; the storage pool is determined by the file placement policy that is in effect when at the time of file creation.
- When the attributes of the file, such as file size or access time, match the rules of a policy that directs GPFS to migrate the data to a different storage pool.

For additional information, refer to:

- "The system storage pool"
- "The system.log storage pool" on page 15
- "User storage pools" on page 15
- "Managing storage pools" on page 16

The system storage pool

The **system** storage pool contains file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks, extended attributes, and so forth.

The **system** storage pool can also contain user data. There is only one **system** storage pool per file system, and it is automatically created when the file system is created.

Important: It is recommended that you use highly-reliable disks and replication for the **system** storage pool because it contains system metadata.

The amount of metadata grows as you add files to the system. Therefore, it is recommended that you monitor the **system** storage pool to ensure that there is always enough space to accommodate growth. The **system** storage pool typically requires a small percentage of the total storage capacity that GPFS manages. However, the percentage required by the **system** storage pool varies depending on your environment. You can monitor the amount of space available in the **system** storage pool with the **mmdf** command. If the available space in the system storage pool begins to run low, you can increase the available space by purging files or adding disks to the system storage pool.

The system.log storage pool

By default the file system recovery log is stored in the system storage pool with file system metadata. The file system recovery log can also be placed in a dedicated pool that is called the **system.log** pool.

This storage pool must be created explicitly. It is highly recommended to only use storage that is as fast or even faster than what is used for the system storage pool. This recommendation is because of the high number of small synchronous data updates made to the recovery log. The block size for the **system.log** pool must be the same as the block size of the system pool.

The file system recovery log will only be stored in one pool.

User storage pools

All user data for a file is stored in the assigned storage pool as determined by your file placement rules.

In addition, file data can be migrated to a different storage pool according to your file management policies. For more information on policies, see “Policies for automating file management” on page 20.

A user storage pool *only* contains the blocks of data (user data, for example) that make up a user file. GPFS stores the data that describes the files, called *file metadata*, separately from the actual file data in the **system** storage pool. You can create one or more user storage pools, and then create policy rules to indicate where the data blocks for a file should be stored.

Tracking file access temperature within a storage pool

A file's access temperature is an attribute for policy that provides a means of optimizing tiered storage. File temperatures are a relative attribute, indicating whether a file is “hotter” or “colder” than the others in its pool. The policy can be used to migrate hotter files to higher tiers and colder files to lower. The access temperature is an exponential moving average of the accesses to the file. As files are accessed the temperature increases; likewise when the access stops the file cools. File temperature is intended to optimize nonvolatile storage, not memory usage; therefore, cache hits are not counted. In a similar manner, only user accesses are counted.

The access counts to a file are tracked as an exponential moving average. An unaccessed file loses a percentage of its accesses each period. The loss percentage and period are set via the configuration variables **fileHeatLossPercent** and **fileHeatPeriodMinutes**. By default, the file access temperature is not tracked. To use access temperature in policy, the tracking must first be enabled. To do this, set the two configuration variables as follows:

fileHeatLossPercent
: The percentage (between 0 and 100) of file access temperature dissipated over the **fileHeatPeriodMinutes** time. The default value is 10.

fileHeatPeriodMinutes
: The number of minutes defined for the recalculation of file access temperature. To turn on tracking, **fileHeatPeriodMinutes** must be set to a nonzero value. The default value is 0.

The following example sets **fileHeatPeriodMinutes** to 1440 (24 hours) and **fileHeatLossPercent** to 10, meaning that unaccessed files will lose 10% of their heat value every 24 hours, or approximately 0.4% every hour (because the loss is continuous and “compounded” geometrically):

```
mmchconfig fileheatperiodminutes=1440,fileheatlosspercent=10
```

Note: If the updating of the file access time (atime) is suppressed or if relative atime semantics are in effect, proper calculation of the file access temperature may be adversely affected.

File access temperature is tracked on a per-cluster basis, not on a per–file system basis.

Use **WEIGHT(FILE_HEAT)** with a policy **MIGRATE** rule to prioritize migration by file temperature. (You can use the **GROUP POOL** rule to define a group pool to be specified as the **TO POOL** target.) See "Policies for automating file management" on page 20.

Managing storage pools

Managing your storage pools includes:
- "Creating storage pools"
- "Changing the storage pool assignment of a disk"
- "Changing the storage pool assignment of a file" on page 17
- "Deleting storage pools" on page 17
- "Listing the storage pools of a file system" on page 17
- "Listing the storage pool of a file" on page 18
- "Listing disks and associated statistics" on page 18
- "Rebalancing files in a storage pool" on page 19
- "Using replication in a storage pool" on page 19

Creating storage pools:

The storage pool to which a disk belongs is an attribute of each disk and is specified as a field in each disk descriptor when the file system is created using the **mmcrfs** command or when disks are added to an existing file system with the **mmadddisk** command. Adding a disk with a new storage pool name in the disk descriptor automatically creates the storage pool.

Storage pool names:
- Must be unique within a file system, but not across file systems.
- Cannot be longer than 255 alphanumeric characters.
- Are case sensitive. **MYpool** and **myPool** are distinct storage pools.

A storage pool is defined by the stanza keyword **pool**; for example:

```
pool=dataPoolA
```

If a storage pool is not specified, the disk is by default assigned to the **system** storage pool.

The **--metadata-block-size** flag on the **mmcrfs** command can be used to create a system pool with a different block size from the user pools. This can be especially beneficial if the default block size is larger than 1 MB. If data and metadata block sizes differ, the system pool must contain only **metadataOnly** disks.

Changing the storage pool assignment of a disk:

Once a disk is assigned to a storage pool, the pool assignment cannot be changed by using either the **mmchdisk** command or the **mmrpldisk** command. You can, however, change the pool to which the disk is assigned.

To move a disk to another pool:

1. Delete the disk from its current pool by issuing the **mmdeldisk** command. This will move the data to the remaining disks in the storage pool.
2. Add the disk to the new pool by issuing the **mmadddisk** command.
3. Rebalance the data across all disks in the new storage pool by issuing the **mmrestripefs -P** command.

Changing the storage pool assignment of a file:

You can change the storage pool that a file is assigned to.

A root user can change the storage pool that a file is assigned to by either:
- Running **mmapplypolicy** with an appropriate set of policy rules.
- Issuing the **mmchattr -P** command.

By default, both of these commands migrate data immediately (this is the same as using the **-I yes** option for these commands). If desired, you can delay migrating the data by specifying the **-I defer** option for either command. Using the defer option, the existing data does not get moved to the new storage pool until either the **mmrestripefs** command or the **mmrestripefile** command are executed. For additional information, refer to:
- "Overview of policies" on page 20
- "Rebalancing files in a storage pool" on page 19

Deleting storage pools:

System storage pools, **system.log** pools and user storage pools have different deletion requirements.

Deleting the **System** storage pool is not allowed. You can delete the **system** storage pool only after you have deleted the file system.

You can delete the **system.log** pool by deleting all the disks in the **system.log** pool. You do not need to run a policy to empty the **system.log** pool first, because the **system.log** pool can only contain log files, and those are automatically migrated to the **System** pool when you delete the **System** pool.

In order to delete a user storage pool, you must delete all its disks using the **mmdeldisk** command. When GPFS deletes the last remaining disk from a user storage pool, the storage pool is also deleted. To delete a storage pool, it must be completely empty. A migration policy along with the **mmapplypolicy** command could be used to do this.

Listing the storage pools of a file system:

To list the storage pools available for a specific file system, issue the **mmlsfs -P** command.

For example, this command:

```
mmlsfs fs1 -P
```

produces output similar to this:

```
flag               value                    description
------------------ ------------------------ -----------------------------------
 -P                system;sp1;sp2           Disk storage pools in file system
```

For file system **fs1**, there are three storage pools: the **system** storage pool and user storage pools named **sp1** and **sp2**.

Listing the storage pool of a file:

To display the assigned storage pool and the name of the fileset that includes the file, issue the **mmlsattr -L** command.

For example, this command:

```
mmlsattr -L myfile
```

produces output similar to this:

```
file name:            myfile
metadata replication: 2 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    sp1
fileset name:         root
snapshot name:
creation Time:        Wed Feb 22 15:16:29 2012
Misc attributes:      ARCHIVE
```

File **myfile** is assigned to the storage pool named **sp1** and is part of the root fileset.

Listing disks and associated statistics:

To list the disks belonging to a storage pool, issue the **mmdf -P** command.

For example, this command:

```
mmdf fs1 -P sp1
```

produces output similar to this:

```
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: sp1 (Maximum disk size allowed is 1.2 TB)
vp4vsdn05            17760256        6 no       yes       11310080 ( 64%)        205200 ( 1%)
vp5vsdn05            17760256        6 no       yes       11311104 ( 64%)        205136 ( 1%)
vp6vsdn05            17760256        6 no       yes       11300352 ( 64%)        206816 ( 1%)
vp7vsdn05            17760256        6 no       yes       11296256 ( 64%)        209872 ( 1%)
vp0vsdn05            17760256        6 no       yes       11293696 ( 64%)        207968 ( 1%)
vp1vsdn05            17760256        6 no       yes       11293184 ( 64%)        206464 ( 1%)
vp2vsdn05            17760256        6 no       yes       11309056 ( 64%)        203248 ( 1%)
vp3vsdn05            17760256        6 no       yes       11269120 ( 63%)        211456 ( 1%)
                -------------                         -------------------- -------------------
(pool total)        142082048                             90382848 ( 64%)       1656160 ( 1%)
```

This example shows that storage pool **sp1** in file system **fs1** consists of eight disks and identifies details for each disk including:

- Name
- Size
- Failure group
- Data type
- Free space

Rebalancing files in a storage pool:

A root user can rebalance file data across all disks in a file system by issuing the **mmrestripefs** command.

Optionally:
- Specifying the **-P** option rebalances only those files assigned to the specified storage pool.
- Specifying the **-p** option rebalances the file placement within the storage pool. For files that are assigned to one storage pool, but that have data in a different pool, (referred to as ill-placed files), using this option migrates their data to the correct pool. (A file becomes "ill-placed" when the **-I defer** option is used during migration of the file between pools.)

Using replication in a storage pool:

To enable data replication in a storage pool, you must make certain that there are at least two failure groups within the storage pool.

This is necessary because GPFS maintains separation between storage pools and performs file replication within each storage pool. In other words, a file and its replica must be in the same storage pool. This also means that if you are going to replicate the entire file system, every storage pool in the file system must have at least two failure groups.

Note: Depending on the configuration of your file system, if you try to enable file replication in a storage pool having only one failure group, GPFS will either give you a warning or an error message.

External storage pools

When you initially create a file, GPFS assigns that file to an internal storage pool. Internal storage pools support various types of online storage. To move data from online storage to offline or near-line storage, you can use external storage pools.

External storage pools use a flexible interface driven by GPFS policy rules that simplify data migration to and from other types of storage such as tape storage. For additional information, refer to "Policies for automating file management" on page 20.

You can define multiple external storage pools at any time using GPFS policy rules. To move data to an external storage pool, the GPFS policy engine evaluates the rules that determine which files qualify for transfer to the external pool. From that information, GPFS provides a list of candidate files and executes the script specified in the rule that defines the external pool. That executable script is the interface to the external application, such as Tivoli Storage Manager (TSM), that does the actual migration of data into an external pool. Using the external pool interface, GPFS gives you the ability to manage information by allowing you to:
1. Move files and their extended attributes onto low-cost near-line or offline storage when demand for the files diminishes.
2. Recall the files, with all of their previous access information, onto online storage whenever the files are needed.

External pool requirements

With external pools, GPFS provides metadata processing and the flexibility of using extended file attributes. The external storage manager is responsible for moving files from GPFS and returning them upon the request of an application accessing the file system. Therefore, when you are using external storage pools, you must use an external file management application such as TSM. The external application is responsible for maintaining the file once it has left the GPFS file system. For example, GPFS policy rules create a list of files that are eligible for migration. GPFS hands that list to TSM which migrates the files to tape and creates a reference file in the file system that has pointers to the tape image.

When a file is requested, it is automatically retrieved from the external storage pool and placed back in an internal storage pool. As an alternative, you can use a GPFS policy rule to retrieve the data in advance of a user request.

The number of external storage pools is only limited by the capabilities of your external application. GPFS allows you to define external storage pools at any time by writing a policy that defines the pool and makes that location known to GPFS. External storage pools are defined by policy rules and initiated by either storage thresholds or use of the **mmapplypolicy** command.

For additional information, refer to "Working with external storage pools" on page 48.

Policies for automating file management

GPFS provides a means to automate the management of files using policies and rules. Properly managing your files allows you to efficiently use and balance your premium and less expensive storage resources.

GPFS supports these policies:
- *File placement policies* are used to automatically place newly created files in a specific storage pool.
- *File management policies* are used to manage files during their lifecycle by moving them to another storage pool, moving them to near-line storage, copying them to archival storage, changing their replication status, or deleting them.

You can create and manage policies and policy rules with both the command line interface and the GUI. In the GUI, navigate to **Files** > **Information Lifecycle Management**.

Overview of policies

A *policy* is a set of rules that describes the life cycle of user data based on the attributes of files. Each rule defines an operation or definition, such as "migrate to a pool and replicate the file."

You can do the following tasks with rules:
- Initial file placement
- File management
- Restoring file data
- Encryption-specific uses. For more information, see the topic *Encryption* in the *IBM Spectrum Scale: Administration and Programming Reference*.
- File compression and decompression. For more information, see the topic *File compression* in the *IBM Spectrum Scale: Administration and Programming Reference*.

When a file is created or restored, the placement policy determines the location of the file's data and assigns the file to a storage pool. All data written to that file is placed in the assigned storage pool.

The placement policy defining the initial placement of newly created files and the rules for placement of restored data must be installed into GPFS with the **mmchpolicy** command. If a GPFS file system does not have a placement policy installed, all the data is stored in the first data storage pool. Only one placement policy can be installed at a time. If you switch from one placement policy to another, or make changes to a placement policy, that action has no effect on existing files. However, newly created files are always placed according to the currently installed placement policy.

The management policy determines file management operations such as migration, deletion, and file compression or decompression.

In order to migrate or delete data, you must use the **mmapplypolicy** command. To compress or decompress data, you can use either the **mmapplypolicy** command with a **MIGRATE** rule or the **mmchattr** command. You can define the file management rules and install them in the file system together with the placement rules. As an alternative, you may define these rules in a separate file and explicitly provide them to **mmapplypolicy** using the **-P** option. In either case, policy rules for placement or migration may be intermixed. Over the life of the file, data can be migrated to a different storage pool any number of times, and files can be deleted or restored.

Note: In a multicluster environment, the scope of the **mmapplypolicy** command is limited to the nodes in the cluster that owns the file system.

Note: File compression or decompression using the **mmapplypolicy** command is not supported on the Windows operating system.

File management rules can also be used to control the space utilization of GPFS online storage pools. When the utilization for an online pool exceeds the specified high threshold value, GPFS can be configured, through user exits, to trigger an event that can automatically start **mmapplypolicy** and reduce the utilization of the pool. Using the **mmaddcallback** command, you can specify a script that will run when such an event occurs. For more information, see the topic *mmaddcallback command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

GPFS performs error checking for file-placement policies in the following phases:
- When you install a new policy, GPFS checks the basic syntax of all the rules in the policy.
- GPFS also checks all references to storage pools. If a rule in the policy refers to a storage pool that does not exist, the policy is not installed and an error is returned.
- When a new file is created, the rules in the active policy are evaluated in order. If an error is detected, GPFS logs an error, skips all subsequent rules, and returns an **EINVAL** error code to the application.
- Otherwise, the first applicable rule is used to store the file data.

Default file placement policy:

When a GPFS file system is first created, the default file placement policy is to assign all files to the **system** storage pool. You can go back to the default policy by running the command:

```
mmchpolicy Device DEFAULT
```

For more information on using GPFS commands to manage policies, see "Managing policies" on page 46.

Policy rules

A *policy rule* is an SQL-like statement that tells GPFS what to do with the data for a file in a specific storage pool if the file meets specific criteria. A rule can apply to any file being created or only to files being created within a specific fileset or group of filesets.

Rules specify conditions that, when true, cause the rule to be applied. Conditions that cause GPFS to apply a rule include:
- Date and time when the rule is evaluated, that is, the current date and time
- Date and time when the file was last accessed
- Date and time when the file was last modified
- Fileset name
- File name or extension
- File size
- User ID and group ID

GPFS evaluates policy rules in order, from first to last, as they appear in the policy. The first rule that matches determines what is to be done with that file. For example, when a client creates a file, GPFS scans the list of rules in the active file placement policy to determine which rule applies to the file. When a rule applies to the file, GPFS stops processing the rules and assigns the file to the appropriate storage pool. If no rule applies, an EINVAL error code is returned to the application.

There are eight types of policy rules that allow you to define specific actions that GPFS will implement on the file data. Each rule has clauses that control candidate selection, namely when the rule is allowed to match a file, what files it will match, the order to operate on the matching files and additional attributes to show for each candidate file. Different clauses are permitted on different rules based upon the semantics of the rule.

Policy rules: Syntax

Policy rules can apply to file placements, group pools, file migrations, file deletions, file exclusions, file lists, file restores, external storage pool definitions, and external list definitions.

The policy rules and their respective syntax diagrams are as follows. For more information about encryption-specific rules, see Chapter 19, “Encryption,” on page 367.

- File placement rules

```
RULE ['RuleName']
  SET POOL 'PoolName'
     [LIMIT (OccupancyPercentage)]
     [REPLICATE (DataReplication)]
     [FOR FILESET ('FilesetName'[,'FilesetName']...)]
     [ACTION (SqlExpression)]
     [WHERE SqlExpression]
```

- Group pool rule; used to define a list of pools that may be used as a pseudo-pool source or destination in either a **FROM POOL** or **TO POOL** clause within another rule

```
RULE ['RuleName'] GROUP POOL ['groupPoolName']
IS 'poolName' [LIMIT(OccupancyPercentage)]
THEN 'poolName2' [LIMIT(n2)]
THEN 'pool-C' [LIMIT(n3)]
THEN ...
```

- File migration rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  MIGRATE    [COMPRESS ({'yes' | 'no'})]
    [FROM POOL 'FromPoolName']
    [THRESHOLD (HighPercentage[,LowPercentage[,PremigratePercentage]])]
    [WEIGHT (WeightExpression)]
  TO POOL 'ToPoolName'
    [LIMIT (OccupancyPercentage)]
    [REPLICATE (DataReplication)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [SHOW (['String'] SqlExpression)]
    [SIZE (numeric-sql-expression)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

| For more information about file compression and decompression, see the topic *File compression* in the
| *IBM Spectrum Scale: Administration and Programming Reference.*

- File deletion rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  DELETE
    [DIRECTORIES_PLUS]
    [FROM POOL 'FromPoolName']
    [THRESHOLD (HighPercentage[,LowPercentage])]
    [WEIGHT (WeightExpression)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [SHOW (['String'] SqlExpression)]
    [SIZE (numeric-sql-expression)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

- File exclusion rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  EXCLUDE
   [DIRECTORIES_PLUS]
   [FROM POOL 'FromPoolName']
   [FOR FILESET ('FilesetName'[,'FilesetName']...)]
   [ACTION (SqlExpression)]
   [WHERE SqlExpression]
```

- File list rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  LIST 'ListName'
    [EXCLUDE]
    [DIRECTORIES_PLUS]
    [FROM POOL 'FromPoolName']
    [THRESHOLD (HighPercentage[,LowPercentage])]
    [WEIGHT (WeightExpression)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [SHOW (['String'] SqlExpression)]
    [SIZE (numeric-sql-expression)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

- File restore rule

```
RULE ['RuleName']
  RESTORE TO POOL 'PoolName'
    [LIMIT (OccupancyPercentage)]
    [REPLICATE (DataReplication)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [ACTION (SqlExpression)]
    [WHERE SqlExpression]
```

- External storage pool definition rule

```
RULE ['RuleName']
  EXTERNAL POOL 'PoolName'
   EXEC 'InterfaceScript'
    [OPTS 'OptionsString ...']
    [ESCAPE '%SpecialCharacters']
    [SIZE sum-number]
```

- External list definition rule

```
RULE ['RuleName']
  EXTERNAL LIST 'ListName'
   EXEC 'InterfaceScript'
    [OPTS 'OptionsString ...']
    [ESCAPE '%SpecialCharacters']
    [THRESHOLD 'ResourceClass']
    [SIZE sum-number]
```

Policy rules: Terms

The terms of policy rules specify conditions for selecting files and operations to perform on files.

The following terms are used in policy rules. Some terms appear in more than one rule:

ACTION (*SqlExpression*)
Specifies an SQL expression that is evaluated only if the other clauses of the rule are satisfied. The action of the *SqlExpression* is completed, and the resulting value of the *SqlExpression* is discarded. In the following example, the rule sets the extended attribute “user.action” to the value “set pool s6” for files that begin with the characters “sp”. These files are assigned to the system pool:

```
rule 's6' set pool 'system' action(setxattr('user.action','set pool s6')) where name like 'sp%'
```

Note: Encryption policies do not support the **ACTION** clause.

COMPRESS ({'yes' | 'no})
Indicates that the file is to be compressed or decompressed. The following rule compresses the files in the pool `datapool` that begin with the string `green%`:

```
RULE 'COMPR1' MIGRATE FROM POOL 'datapool' COMPRESS('yes') WHERE NAME LIKE 'green%'
```

| For more information about file compression and decompression, see the topic *File compression* in the
| *IBM Spectrum Scale: Administration and Programming Reference*.

DIRECTORIES_PLUS
Indicates that non-regular file objects (directories, symbolic links, and other items) must be included in the list. If not specified, only ordinary data files are included in the candidate lists.

DELETE
Identifies a file deletion rule. A file that matches this rule becomes a candidate for deletion.

ESCAPE '*%SpecialCharacters*'
Specifies that path names and the **SHOW**('*string*') expressions within the associated file lists are encoded with a scheme based on RFC3986 URI percent encoding.

Compare the two following rules:

```
RULE 'xp' EXTERNAL POOL 'pool-name' EXEC 'script-name' ESCAPE '%'
RULE 'xl' EXTERNAL LIST 'list-name' EXEC 'script-name' ESCAPE '%/+@#'
```

Both rules specify that all characters except the "unreserved" characters in the set `a-zA-Z0-9-_.~` are encoded as `%XX`, where `XX` comprises two hexadecimal digits.

However, the GPFS **ESCAPE** syntax adds to the set of "unreserved" characters. In the first rule, the syntax `ESCAPE '%'` specifies a rigorous RFC3986 encoding. Under this rule, a path name such as `/root/directory/@abc+def#ghi.jkl` appears in a file list in the following format:

```
%2Froot%2Fdirectory%2F%40abc%2Bdef%23ghi.jkl
```

In the second rule, the syntax `ESCAPE '%/+@#'` specifies that none of the characters in set `/+@#` are escaped. Under this rule, the same path name appears in a file list in the following format:

```
/root/directory/@abc+def#ghi.jkl
```

If you omit the **ESCAPE** clause, the newline character is escaped as `'\n'`, and the backslash character is escaped as `'\\'`; all other characters are presented as is, without further encoding.

EXCLUDE
Identifies a file exclusion rule.

RULE '*x*' EXCLUDE
A file that matches this form of the rule is excluded from further consideration by any **MIGRATE** or **DELETE** rules that follow.

RULE '*rule-name*' LIST '*listname-y*' EXCLUDE
A file that matches this form of the rule is excluded from further consideration by any **LIST** rules that name the same *listname-y*.

EXEC '*InterfaceScript*'
Specifies an external program to be invoked to pass requests to an external storage management application. *InterfaceScript* must be a fully qualified path name to a user-provided script or program that supports the commands described in "User-provided program for managing external pools" on page 49.

EXTERNAL LIST *ListName*
Defines an external list. This rule does not match files. It provides the binding between the lists that are generated with regular **LIST** rules with a matching *ListName* and the external program that you want to run with these lists as input.

EXTERNAL POOL *PoolName*
Defines an external storage pool. This rule does not match files but defines the binding between the policy language and the external storage manager that implements the external storage.

FOR FILESET ('*FilesetName*'[,'*FilesetName*']...)
Specifies that the rule applies only to files within the specified filesets.

FROM POOL *FromPoolName*
Specifies the name of the source pool from which files are candidates for migration.

GROUP POOL *PoolName*
Defines a group pool. This rule supports the concept of distributing data files over several GPFS disk pools.

Optionally, a **LIMIT**, specified as an occupancy percentage, can be specified for each disk pool; if not specified, the limit defaults to 99%. The **THEN** keyword signifies that disk pools that are specified before a **THEN** keyword are preferred over disk pools that are specified after. When a pool that is defined by a **GROUP POOL** rule is the **TO POOL** target of a **MIGRATE** rule, the selected files are distributed among the disk pools that comprise the group pool. Files of highest weight are put into the most preferred disk pool up to the occupancy limit for that pool. If more files must be migrated, they are put into the second most preferred pool up to the occupancy limit for that pool. Again, files of highest weight are selected.

If you specify a file that is defined by a **GROUP POOL** rule in a **FROM POOL** clause, the clause matches any file in any of the disk pools in the group pool.

You can “repack” a group pool by **WEIGHT**. Migrate files of higher weight to preferred disk pools by specifying a group pool as both the source and the target of a **MIGRATE** rule.

```
rule 'grpdef' GROUP POOL 'gpool' IS 'ssd' LIMIT(90) THEN 'fast' LIMIT(85) THEN 'sata'
rule 'repack' MIGRATE FROM POOL 'gpool' TO POOL 'gpool' WEIGHT(FILE_HEAT)
```

See “Tracking file access temperature within a storage pool” on page 15.

LIMIT (*OccupancyPercentage***)**
Limits the creation of data in a storage pool. GPFS does not migrate a file into a pool if doing so
| exceeds the occupancy percentage for the pool. If you do not specify an occupancy percentage for a
| pool, the default value is 99%. See “Phase two: Choosing and scheduling files” on page 40.

You can specify *OccupancyPercentage* as a floating point number, as in the following example:

```
RULE 'r' RESTORE to pool 'x' limit(8.9e1)
```

For testing or planning purposes, and when you use the **mmapplypolicy** command with the **-I defer** or **-I test** option, you can specify a **LIMIT** larger than 100%.

The limit clause does not apply when the target **TO POOL** is a **GROUP POOL**. The limits that are specified in the rule that defines the target **GROUP POOL** govern the action of the **MIGRATE** rule.

LIST *ListName*
Identifies a file list generation rule. A file can match more than one list rule but appears in a list only once. *ListName* provides the binding to an **EXTERNAL LIST** rule that specifies the executable program to call when the generated list is processed.

MIGRATE
Identifies a file migration rule. A file that matches this rule becomes a candidate for migration to the pool specified by the **TO POOL** clause.

OPTS '*OptionsString ...***'**
Specifies optional parameters to be passed to the external program defined with the **EXEC** clause. *OptionsString* is not interpreted by the GPFS policy engine.

REPLICATE (*DataReplication***)**
Overrides the default data replication factor. This value must be specified as 1, 2, or 3.

RESTORE TO POOL *PoolName*
Identifies a file restore rule. When you restore a file with the **gpfs_fputattrswithpathname()** subroutine, you can use this rule to match files against their saved attributes rather than the current file attributes. This rule also applies to a command that uses that subroutine, such as the Tivoli Storage Manager command **dsmc restore**.

RULE ['*RuleName*']
: Initiates the rule statement. *RuleName* identifies the rule and is used in diagnostic messages.

SET POOL *PoolName*
: Identifies an initial file placement rule. *PoolName* specifies the name of the storage pool where all files that match the rule criteria is placed.

SHOW (['*String*'] *SqlExpression*)
: Inserts the requested information (the character representation of the evaluated SQL expression *SqlExpression*) into the candidate list that is created by the rule when it deals with external storage pools. *String* is a literal value that gets echoed back.

  This clause has no effect in matching files but can be used to define other attributes to be exported with the candidate file lists.

SIZE (*numeric-sql-expression*)
: Is an optional clause of any **MIGRATE**, **DELETE**, or **LIST** rules that are used for choosing candidate files. *numeric-sql-expression* specifies the size of the file to be used when in calculating the total amount of data to be passed to a user script. The default is **KB_ALLOCATED**.

SIZE *sum-number*
: Is an optional clause of the **EXTERNAL POOL** and **EXTERNAL LIST** rules. *sum-number* limits the total number of bytes in all of the files named in each list of files passed to your EXEC 'script'. If a single file is larger than *sum-number*, it is passed to your EXEC 'script' as the only entry in a "singleton" file list.

  Specify *sum-number* as a numeric constant or a floating-point value.

  Note: The value of *sum-number* is in kilobytes.

THRESHOLD (*HighPercentage*[,*LowPercentage*[,*PremigratePercentage*]])
: Controls migration and deletion based on the percent of assigned pool storage that is occupied.

  HighPercentage
  : Indicates that the rule is to be applied only if the occupancy percentage of the current pool of the file is greater than or equal to the *HighPercentage* value. Specify a nonnegative integer in the range 0-100.

  LowPercentage
  : Indicates that **MIGRATE** and **DELETE** rules are to be applied until the occupancy percentage of the current pool of the file is reduced to less than or equal to the *LowPercentage* value. Specify a nonnegative integer in the range 0-100. The default is 0%.

  PremigratePercentage
  : Defines an occupancy percentage of a storage pool that is below the lower limit. Files that lie between the lower limit *LowPercentage* and the pre-migrate limit *PremigratePercentage* are copied and become dual-resident in both the internal GPFS storage pool and the designated external storage pool. This option allows the system to free up space quickly by deleting pre-migrated files if the pool becomes full. Specify a nonnegative integer in the range 0 to *LowPercentage*. The default is the same value as *LowPercentage*.

    Notes:
    1. *Percentage* values can be specified as numeric constants or floating-point values.
    2. This option applies only when you migrate to the external storage pool.
    3. This option does not apply when the current rule operates on one group pool.

  THRESHOLD (*ResourceClass*)
  : Specifies the type of capacity-managed resources that are associated with *ListName*. The following values are valid:

FILESET_QUOTAS
: Indicates that the **LIST** rule must use the occupancy percentage of the “hard limit” fileset quota per the **mmlsquota** and **mmedquota** commands.

FILESET_QUOTA_SOFT
: Indicates that the **LIST** rule must use the occupancy percentage of the “soft limit” fileset quota per the **mmlsquota** and **mmedquota** commands.

GROUP_QUOTAS
: Indicates that the **LIST** rule must use the occupancy percentage of the “hard limit” group quota per the **mmlsquota** and **mmedquota** commands.

GROUP_QUOTA_SOFT
: Indicates that the **LIST** rule must use the occupancy percentage of the “soft limit” group quota per the **mmlsquota** and **mmedquota** commands.

POOL_CAPACITIES
: Indicates that the **LIST** rule uses the occupancy percentage of the pool when it applies the threshold rule. This value is the default value. This value is used if the threshold is not specified in the **EXTERNAL LIST** rule but appears in the**LIST** rule.

USER_QUOTAS
: Indicates that the **LIST** rule uses the occupancy percentage of the “hard limit” user quota per the **mmlsquota** and **mmedquota** commands.

USER_QUOTA_SOFT
: Indicates that the **LIST** rule uses the occupancy percentage of the “soft limit” user quota per the **mmlsquota** and **mmedquota** commands.

Note: This option does not apply when the current rule operates on one group pool.

For more detail on how **THRESHOLD** can be used to control file migration and deletion, see “Phase one: Selecting candidate files” on page 39 and “Pre-migrating files with external storage pools” on page 52.

TO POOL *ToPoolName*
: Specifies the name of the storage pool to which all the files that match the rule criteria are migrated. This phrase is optional if the **COMPRESS** keyword is specified.

WEIGHT (*WeightExpression***)**
: Establishes an order on the matching files. Specifies an SQL expression with a numeric value that can be converted to a double-precision floating point number. The expression can refer to any of the file attributes and can include any constants and any of the available SQL operators or built-in functions.

WHEN (*TimeBooleanExpression***)**
: Specifies an SQL expression that evaluates to **TRUE** or **FALSE**, depending only on the SQL built-in variable **CURRENT_TIMESTAMP**. If the **WHEN** clause is present and *TimeBooleanExpression* evaluates to **FALSE**, the rule is skipped.

The **mmapplypolicy** command assigns the **CURRENT_TIMESTAMP** when it begins processing. It uses either the actual Coordinated Universal Time date and time or the date specified with the **-D** option.

WHERE *SqlExpression*
: Specifies an SQL expression that can reference file attributes as SQL variables, functions, and operators. Some attributes are not available to all rules. Compares the file attributes specified in the rule with the attributes of the file that is created.

SqlExpression must be an expression that evaluates to **TRUE** or **FALSE**, but can be any combination of standard SQL syntax expressions, including built-in functions.

Omitting the **WHERE** clause entirely is equivalent to writing **WHERE TRUE**. The **WHERE** clause must be the last clause of the rule.

SQL expressions for policy rules

A number of the available clauses in the GPFS policy rules utilize SQL expressions.

You can reference different file attributes as SQL variables and combine them with SQL functions an operators. Depending on the clause, the SQL expression must evaluate to either **TRUE** or **FALSE**, a numeric value, or a character string. Not all file attributes are available to all rules.

File attributes in SQL expressions:

SQL expressions can include file attributes that specify certain clauses.

The following file attributes can be used in SQL expressions specified with the **WHERE**, **WEIGHT**, and **SHOW** clauses:

`ACCESS_TIME`
: Specifies an SQL time stamp value for the date and time that the file was last accessed (POSIX atime). See **EXPIRATION_TIME**.

`BLOCKSIZE`
: Specifies the size, in bytes, of each block of the file.

`CHANGE_TIME`
: Specifies an SQL time stamp value for the date and time that the file metadata was last changed (POSIX ctime).

`CLONE_DEPTH`
: Specifies the depth of the clone tree for the file.

`CLONE_IS_PARENT`
: Specifies whether the file is a clone parent.

`CLONE_PARENT_FILESETID`
: Specifies the fileset ID of the clone parent. The fileset ID is available only if **CLONE_PARENT_IS_SNAP** is a nonzero value.

`CLONE_PARENT_INODE`
: Specifies the inode number of the clone parent, or **NULL** if it is not a file clone.

`CLONE_PARENT_IS_SNAP`
: Specifies whether the clone parent is in a snapshot.

`CLONE_PARENT_SNAP_ID`
: Specifies the snapshot ID of the clone parent. The snapshot ID is available only if **CLONE_PARENT_IS_SNAP** is a nonzero value.

`CREATION_TIME`
: Specifies an SQL time stamp value that is assigned when a file is created.

`DEVICE_ID`
: Specifies the ID of the device that contains the directory entry.

`DIRECTORY_HASH`
: Can be used to group files within the same directory.

  DIRECTORY_HASH is a function that maps every **PATH_NAME** to a number. All files within the same directory are mapped to the same number and deeper paths are assigned to larger numbers.

  DIRECTORY_HASH uses the following functions:

  `CountSubstr(`*`BigString`*`,`*`LittleString`*`)`
  : Counts and returns the number of occurrences of *LittleString* in *BigString*.

  `HashToFloat(`*`StringValue`*`)`
  : Is a hash function that returns a quasi-random floating point number ≥ 0 and < 1, whose value

depends on a string value. Although the result might appear random, **HashToFloat**(*StringValue*) always returns the same floating point value for a particular string value.

The following rule lists the directory hash values for three directories:

```
RULE 'y' LIST 'xl' SHOW(DIRECTORY_HASH)
LIST 'xl' /abc/tdir/randy1 SHOW(+3.49449638091027E+000)
LIST 'xl' /abc/tdir/ax     SHOW(+3.49449638091027E+000)
LIST 'xl' /abc/tdir/mmPolicy.8368.765871DF/mm_tmp/PWL.12 SHOW(+5.21282524359412E+000)
LIST 'xl' /abc/tdir/mmPolicy.31559.1E018912/mm_tmp/PWL.3 SHOW(+5.10672733094543E+000)
LIST 'xl' /abc/tdir/mmPolicy.31559.1E018912/mm_tmp/PWL.2 SHOW(+5.10672733094543E+000)
```

The following rule causes files within the same directory to be grouped and processed together during deletion. Grouping the files can improve the performance of GPFS directory-locking and caching.

```
RULE 'purge' DELETE WEIGHT(DIRECTORY_HASH) WHERE (deletion-criteria)
```

EXPIRATION_TIME
Specifies the expiration time of the file, expressed as an SQL time-stamp value. If the expiration time of a file is not set, its expiration time is SQL NULL. You can detect such files by checking for "EXPIRATION_TIME IS NULL".

Remember the following points:

- EXPIRATION_TIME is tracked independently from ACCESS_TIME and both values are maintained for immutable files.
- Expiration time and indefinite retention are independent attributes. You can change the value of either one without affecting the value of the other.

FILE_HEAT
Specifies the heat of the file based on the file access time and access size. For more information, see `/usr/lpp/mmfs/samples/ilm/README`.

FILE_SIZE
Specifies the current size or length of the file, in bytes.

FILESET_NAME
Specifies the fileset where the path name for the files is located, or is to be created.

Note: Using the **FOR FILESET** clause has the same effect and is more efficient to evaluate.

GENERATION
Specifies a number that is incremented whenever an INODE number is reused.

GROUP_ID
Specifies the numeric group ID of the file's group.

INODE
Specifies the file's inode number.

KB_ALLOCATED
Specifies the number of kilobytes of disk space allocated for the file data.

MODE
Displays the type and permission bits of a file as a 10-character string. The string has the same format as the first 10 characters of the output from the UNIX **ls -l** command. For example, **-rwxr-xr-x** is the **MODE** string of a file that can be read or executed by its owner, its group, or any user, but written only by its owner.

The first character of the **MODE** attributes displays the file type. The following values are supported:

d Directory.

l Link.

c Character device.

b Block device.

p Pipe.

s Socket.

? Some other attribute, or unknown.

⌂

MISC_ATTRIBUTES
Specifies various miscellaneous file attributes. The value is a string of characters that are defined as follows:

a The file is appendOnly.

A Archive.

c The file is selected to be compressed.

D Directory. To match all directories, you can use %D% as a wildcard character.

F Regular data file.

I Some data blocks might be ill-placed.

J Some data blocks might be ill-replicated.

K Some data blocks might be ill-compressed.

L Symbolic link.

M Co-managed.

2 Data blocks are replicated.

o Offline.

O Other (not F, D, nor L). For example, a device or named pipe.

P AFM

R Read-only.

u AFM cached-complete flag.

U The file is **trunc**-managed.

V Read-managed.

W Write-managed.

X Immutability.

Y Indefinite retention.

MODIFICATION_SNAPID
Specifies the integer ID of the snapshot after which the file was last changed. The value is normally derived with the **SNAP_ID()** built-in function that assigns integer values to GPFS snapshot names. This attribute allows policy rules to select files that are modified after a snapshot image is taken.

MODIFICATION_TIME
Specifies an SQL time stamp value for the date and time that the file data was last modified (POSIX mtime).

NAME
Specifies the name of a file.

NLINK
Specifies the number of hard links to the file.

PATH_NAME
Specifies a path for the file; the path includes the name of the file.

POOL_NAME
Specifies the storage pool where the file data is located.

Note: Using the **FROM POOL** clause has the same effect and is often preferable.

RDEVICE_ID
Specifies the device type for a device.

USER_ID
Specifies the numeric user ID of the owner of the file.

Notes:

1. When file attributes are referenced in initial placement rules, only the following attributes are valid: **FILESET_NAME**, **GROUP_ID**, **NAME**, and **USER_ID**. The placement rules, like all rules with a clause, might also reference the current date and current time and use them to control matching.
2. When file attributes are used for restoring files, the attributes correspond to the attributes at the time of the backup, not to the current restored file.
3. For SQL expressions, if you want to show any of these attribute fields as strings (for example, **FILE_HEAT**), use **SHOW('[FILE_HEAT]')** rather than **SHOW('FILE_HEAT')**, as the latter is expanded.
4. All date attributes are evaluated in Coordinated Universal Time (a time standard abbreviated as UTC).

Using built-in functions:

With GPFS, you can use built-in functions in comparison predicates, between predicates, in predicates, like predicates, mathematical value expressions, and boolean, string and numeric literals.

These functions are organized into the following categories:

- "Extended attribute functions"
- "String functions" on page 35
- "Numerical functions" on page 37
- "Date and time functions" on page 37

Extended attribute functions:

You can use these functions to support access to the extended attributes of a file, and to support conversion of values to the supported SQL data types.

The following attribute functions can be used:

GetXattrs(*pattern*,*prototype*)
Returns extended attribute **key=value** pairs of a file for all extended attributes whose keys that match *pattern*. The **key=value** pairs are returned in the format specified by *prototype*.

If the value specified for *pattern* is `'*'` or empty then all keys are matched.

The *prototype* is a character string representing the format of a typical **key=value** pair. The *prototype* allows the user to specify which characters will be used to quote values, escape special code points, separate the key and value, and separate each **key=value** pair.

Some examples of the *prototype* argument include:

```
key~n=value^n,       # specify the escape characters
hexkey=hexvalue,     # specify either or both as hexadecimal values
"key\n"="value\n",   # specify quotes on either or both
key:"value^n";       # specify alternatives to = and ,
k:"v^n";             # allow key and value to be abbreviated
```

```
key,                # specify keys only
"value~n";          # specify values only
key='value~n'&      # alternative quoting character
key=value           # do not use a ',' separator; use space instead
```

You may omit the last or both arguments. The defaults are effectively **GetXattrs('*','key^n=hexvalue,')**.

The **GetXattrs** function returns an empty string for files that have no extended attributes with keys that match *pattern*.

The **GetXattrs** function is supported by the **mmapplypolicy** command, but it might return **NULL** in other contexts.

SetBGF(*BlockGroupFactor*)
Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works if **--allow-write-affinity** is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

SetWAD(*WriteAffinityDepth*)
Specifies the allocation policy to be used. This option only works if **--allow-write-affinity** is set for the data pool. This applies only to a new data block layout; it does not migrate previously existing data blocks.

SetWADFG("*WadfgValueString*")
Indicates the range of nodes (in a shared nothing architecture) where replicas of blocks in the file are to be written. You use this parameter to determine the layout of a file in the cluster so as to optimize the typical access patterns of your applications. This applies only to a new data block layout; it does not migrate previously existing data blocks.

"*WadfgValueString*" is a semicolon-separated string identifying one or more failure groups in the following format:

```
FailureGroup1[;FailureGroup2[;FailureGroup3]]
```

where each *FailureGroupx* is a comma-separated string identifying the rack (or range of racks), location (or range of locations), and node (or range of nodes) of the failure group in the following format:

```
Rack1{:Rack2{:...{:Rackx}}},Location1{:Location2{:...{:Locationx}}},ExtLg1{:ExtLg2{:...{:ExtLgx}}}
```

For example, the following value

```
1,1,1:2;2,1,1:2;2,0,3:4
```

means that the first failure group is on rack 1, location 1, extLg 1 or 2; the second failure group is on rack 2, location 1, extLg 1 or 2; and the third failure group is on rack 2, location 0, extLg 3 or 4.

If the end part of a failure group string is missing, it is interpreted as 0. For example, the following are interpreted the same way:

```
2
2,0
2,0,0
```

Notes:

1. Only the end part of a failure group string can be left off. The missing end part may be the third field only, or it may be both the second and third fields; however, if the third field is provided, the second field must also be provided. The first field must *always* be provided. In other words, every comma must both follow and precede a number; therefore, *none* of the following are valid:

   ```
   2,0,
   2,
   ,0,0
   0,,0
   ,,0
   ```

2. Wildcard characters (*) are supported in these fields.

Here is an example of using setBGF, setWAD, and setWADFG:

```
RULE 'bgf' SET POOL 'pool1' WHERE NAME LIKE '%' AND setBGF(128) AND setWAD(1) AND setWADFG(1,0,1;2,0,1;3,0,1)
```

This rule has the same effect as the following command:

```
mmchattr --block-group-factor 128 --write-affinity-depth 1 --write-affinity-failuregroup "1,0,1;2,0,1;3,0,1"  test
```

After installing this policy, a newly created file will have the same values for these three extended attributes as it would if **mmchattr** were used to set them:

```
(06:29:11) hs22n42:/sncfs # mmlsattr -L test
file name:            test
metadata replication: 3 max 3
data replication:     3 max 3
immutable:            no
appendOnly:           no
flags:
storage pool name:    system
fileset name:         root
snapshot name:
Block group factor:  128                                         -----------------gpfs.BGF
Write affinity depth:  1                                         -----------------gpfs.WAD
Write Affinity Depth Failure Group(FG) Map for copy:1 1,0,1      -----------------gpfs.WADFG
Write Affinity Depth Failure Group(FG) Map for copy:2 2,0,1
Write Affinity Depth Failure Group(FG) Map for copy:3 3,0,1
creation time:        Sat Jun  8 06:28:50 2013
Misc attributes:      ARCHIVE
```

SetXattr('*ExtendedAttributeName*', '*ExtendedAttributeValue*')
This function sets the value of the specified extended attribute of a file.

Successful evaluation of **SetXattr** in a policy rule returns the value **TRUE** and sets the named extended attribute to the specified value for the file that is the subject or object of the rule. This function is effective for policy rules (like **MIGRATE** and **LIST**) that are evaluated by **mmapplypolicy** and for the policy placement rule, **SET POOL**, when a data file is about to be created.

XATTR(*extended-attribute-name* [, *start* [, *length*]])
Returns the value of a substring of the extended attribute that is named by its argument as an SQL VARCHAR value, where:

extended-attribute-name
Specifies any SQL expression that evaluates to a character string value. If the named extended attribute does not exist, **XATTR** returns the special SQL value **NULL**.

Note: In SQL, the expression **NULL ||** *AnyValue* yields **NULL**. In fact, with a few exceptions, the special SQL value of **NULL** "propagates" throughout an SQL expression, to yield **NULL**. A notable exception is that *(expression)* **IS NULL** always yields either **TRUE** or **FALSE**, never **NULL**.

For example, if you wish to display a string like **_NULL_** when the value of the extended attribute of a file is **NULL** you will need to code your policy rules file like this:

```
define(DISPLAY_NULL,[COALESCE($1,'_NULL_')])
rule external list 'a' exec ''
rule list 'a' SHOW(DISPLAY_NULL(xattr('user.marc')) || ' and ' || DISPLAY_NULL(xattr('user.eric')))
```

Here is an example execution, where either or both of the values of the two named extended attributes may be **NULL**:

```
mmapplypolicy /gig/sill -P /ghome/makaplan/policies/display-null.policy -I test -L 2
   ...
WEIGHT(inf) LIST 'a' /gg/sll/cc SHOW(_NULL_ and _NULL_)
WEIGHT(inf) LIST 'a' /gg/sll/mm SHOW(yes-marc and _NULL_)
WEIGHT(inf) LIST 'a' /gg/sll/bb SHOW(_NULL_ and yes-eric)
WEIGHT(inf) LIST 'a' /gg/sll/tt SHOW(yes-marc and yes-eric)
```

GPFS/Policy/SQL is a subset of standard ISO/ANSI SQL, with additional extensions and modifications to facilitate GPFS/ILM. Regarding the **NULL** value, GPFS/Policy/SQL supports the "unknown value" meaning of **NULL**.

start
: Is the optional starting position within the extended attribute value. The default is 1.

length
: Is the optional length, in bytes, of the extended attribute value to return. The default is the number of bytes from the start to the end of the extended attribute string.

Note: **XATTR** (*name*,*i*,*k*) == **SUBSTR**(**XATTR**(*name*),*i*,*k*).

Some extended attribute values represent numbers or timestamps as decimal or binary strings. Use the **TIMESTAMP**, **XATTR_FLOAT**, or **XATTR_INTEGER** function to convert extended attributes to SQL numeric or timestamp values:

XATTR_FLOAT(*extended-attribute-name* **[,** *start* **[,** *length*, **[,** *conversion_option***]]])**
Returns the value of a substring of the extended attribute that is named by its argument, converted to an SQL double floating-point value, where:

extended-attribute-name
Specifies any SQL expression that evaluates to a character string value. If the named extended attribute does not exist, **XATTR** returns the special SQL value **NULL**.

start
Is the optional starting position within the extended attribute value. The default is 1.

length
Is the optional length, in bytes, of the extended attribute value to return. The default is the number of bytes from the start to the end of the extended attribute string. You can specify length as -1 to reach from the start to the end of the extended attribute string.

conversion_option
Specifies how the bytes are to be converted to a floating-point value. Supported options include:

- BIG_ENDIAN_DOUBLE or BD - a signed binary representation, IEEE floating, sign + 11 bit exponent + fraction. This is the default when executing on a "big endian" host OS, such as AIX on PowerPC®.
- BIG_ENDIAN_SINGLE or BS - IEEE floating, sign + 8-bit exponent + fraction.
- LITTLE_ENDIAN_DOUBLE or LD - bytewise reversed binary representation. This is the default when executing on a "little endian" host OS, such as Linux on Intel x86.
- LITTLE_ENDIAN_SINGLE or LS - bytewise-reversed binary representation.
- DECIMAL - the conventional SQL character string representation of a floating-point value.

Notes:

1. Any prefix of a conversion name can be specified instead of spelling out the whole name. The first match against the list of supported options is used; for example, L matches LITTLE_ENDIAN_DOUBLE.
2. If the extended attribute does not exist, the selected substring has a length of 0, or the selected bytes cannot be converted to a floating-point value, the function returns the special SQL value **NULL**.

XATTR_INTEGER(*extended-attribute-name* **[,** *start* **[,** *length*, **[,** *conversion_option***]]])**
Returns the value of (a substring of) the extended attribute named by its argument, converted to a SQL LARGEINT value, where.

extended-attribute-name
Specifies any SQL expression that evaluates to a character string value. If the named extended attribute does not exist, **XATTR** returns the special SQL value **NULL**.

start
Is the optional starting position within the extended attribute value. The default is 1.

length
Is the optional length, in bytes, of the extended attribute value to return. The default is the number of bytes from the start to the end of the extended attribute string. You can specify length as -1 to reach from the start to the end of the extended attribute string.

conversion_option
Specifies how the bytes are to be converted to a LARGEINT value. Supported options include:

- BIG_ENDIAN - a signed binary representation, most significant byte first. This is the default when executing on a "big endian" host OS, such as AIX on PowerPC.
- LITTLE_ENDIAN - bytewise reversed binary representation. This is the default when executing on a "little endian" host OS, such as Linux on Intel x86.
- DECIMAL - the conventional SQL character string representation of an integer value.

Notes:

1. Any prefix of a conversion name can be specified instead of spelling out the whole name (B, L, or D, for example).
2. If the extended attribute does not exist, the selected substring has a length of 0, or the selected bytes cannot be converted to a LARGEINT value, the function returns the special SQL value **NULL**. For example:

```
XATTR_INTEGER('xyz.jim',5,-1,'DECIMAL')
```

String functions:

You can use these string-manipulation functions on file names and literal values.

Important tips:

1. You must enclose strings in single-quotation marks.
2. You can include a single-quotation mark in a string by using two single-quotation marks. For example, **'a''b'** represents the string **a'b**.

CHAR(*expr***[,** *length***])**
Returns a fixed-length character string representation of its *expr* argument, where:

expr
Can be any data type.

length
If present, must be a literal, integer value.

The resulting type is **CHAR** or **VARCHAR**, depending upon the particular function called.

The string that **CHAR** returns is padded with blanks to fill the *length* of the string. If *length* is not specified, it defaults to a value that depends on the type of the argument (*expr*).

Note: The maximum length of a **CHAR** (fixed length string) value is 255 bytes. The result of evaluating an SQL expression whose result is type **CHAR** may be truncated to this maximum length.

CONCAT(*x***,***y***)**
Concatenates strings *x* and *y*.

HEX(*x***)**
Converts an integer *x* into hexadecimal format.

LENGTH(*x***)**
Determines the length of the data type of string *x*.

LOWER(*x*)
: Converts string *x* into lowercase.

REGEX(*String*,'*Pattern*')
: Returns **TRUE** if the pattern matches, **FALSE** if it does not. *Pattern* is a Posix extended regular expression.

Note: The policy SQL parser normally performs M4 macro preprocessing with square brackets set as the quote characters. Therefore, it is recommended that you add an extra set of square brackets around your **REGEX** pattern string; for example:

```
...WHERE REGEX(name,['^[a-z]*$'])  /* only accept lowercase alphabetic file names */
```

The following SQL expression:

```
NOT REGEX(STRING_VALUE,['^[^z]*$|^[^y]*$|^[^x]*$|[abc]'])
```

can be used to test if STRING_VALUE contains *all* of the characters x, y, and z, in any order, and *none* of the characters a, b, or c.

REGEXREPLACE(*string*,*pattern*,*result-prototype-string*)
: Returns a character string as *result-prototype-string* with occurrences of *i* (where *i* is 0 through 9) replaced by the substrings of the original string that match the *i*th parenthesis delimited parts of the pattern string. For example:

```
REGEXREPLACE('speechless',['([^aeiou]*)([aeiou]*)(.*)'],['last=\3. middle=\2. first=\1.'])
```

returns the following:

```
'last=chless. middle=ee. first=sp.'
```

When *pattern* does not match *string*, **REGEXREPLACE** returns the value **NULL**.

When a **\0** is specified in the *result-prototype-string*, it is replaced by the substring of *string* that matches the entire *pattern*.

SUBSTR(*x*,*y*,*z*)
: Extracts a portion of string *x*, starting at position *y*, optionally for *z* characters (otherwise to the end of the string). This is the short form of **SUBSTRING**. If *y* is a negative number, the starting position is counted from the end of the string; for example, **SUBSTR('ABCDEFGH',-3,2)** == 'FG'.

Note: Do not confuse **SUBSTR** with **substr**. **substr** is an m4 built-in macro function.

SUBSTRING(*x* FROM *y* FOR *z*)
: Extracts a portion of string *x*, starting at position *y*, optionally for *z* characters (otherwise to the end of the string).

UPPER(*x*)
: Converts the string *x* into uppercase.

VARCHAR(*expr* [, *length*])
: Returns a varying-length character string representation of a character string, date/time value, or numeric value, where:

expr
: Can be any data type.

length
: If present, must be a literal, integer value.

The resulting type is **CHAR** or **VARCHAR**, depending upon the particular function called. Unlike **CHAR**, the string that the **VARCHAR** function returns is not padded with blanks.

Note: The maximum length of a **VARCHAR**(variable length string) value is 8192 bytes. The result of evaluating an SQL expression whose result is type **VARCHAR** may be truncated to this maximum length.

Numerical functions:

You can use numeric-calculation functions to place files based on either numeric parts of the file name, numeric parts of the current date, or UNIX-client user IDs or group IDs.

These functions can be used in combination with comparison predicates and mathematical infix operators (such as addition, subtraction, multiplication, division, modulo division, and exponentiation).

INT(*x*)
Converts number *x* to a whole number, rounding up fractions of .5 or greater.

INTEGER(*x*)
Converts number *x* to a whole number, rounding up fractions of .5 or greater.

MOD(*x*,*y*)
Determines the value of *x* taken modulo *y* (*x* % *y*).

Date and time functions:

You can use these date-manipulation and time-manipulation functions to place files based on when the files are created and the local time of the GPFS node serving the directory where the file is being created.

CURRENT_DATE
Determines the current date on the GPFS server.

CURRENT_TIMESTAMP
Determines the current date and time on the GPFS server.

DAYOFWEEK(*x*)
Determines the day of the week from date or timestamp *x*. The day of a week is from 1 to 7 (Sunday is 1).

DAYOFYEAR(*x*)
Determines the day of the year from date *x*. The day of a year is a number from 1 to 366.

DAY(*x*)
Determines the day of the month from date or timestamp *x*.

DAYS(*x*)
Determines the number of days between date or timestamp *x* and 0001-01-01.

DAYSINMONTH(*x*)
Determines the number of days in the month of date *x*.

DAYSINYEAR(*x*)
Determines the day of the year of date *x*.

HOUR(*x*)
Determines the hour of the day (a value from 0 to 23) of timestamp *x*.

MINUTE(*x*)
Determines the minute from timestamp *x*.

MONTH(*x*)
Determines the month of the year from date or timestamp *x*.

QUARTER(*x*)
Determines the quarter of year from date *x*. Quarter values are the numbers 1 through 4. For example, January, February, and March are in quarter 1.

SECOND(*x*)
Returns the seconds portion of timestamp *x*.

TIMESTAMP(*sql-numeric-value*) or TIMESTAMP(*sql-character-string-value*)
Accepts any numeric value. The numeric value is interpreted as the number of seconds since January 1, 1970 (the standard UNIX epoch) and is converted to an SQL TIMESTAMP value.

Signed 64-bit LARGEINT argument values are supported. Negative argument values cause **TIMESTAMP** to convert these values to timestamps that represent years before the UNIX epoch.

This function also accepts character strings of the form *YYYY-MM-DD HH:MM:SS*. A hyphen (-) or an at sign (@) might appear instead of the blank between the date and the time. The time can be omitted. An omitted time defaults to 00:00:00. The *:SS* field can be omitted, which defaults to 00.

WEEK(*x*)
Determines the week of the year from date *x*.

YEAR(*x*)
Determines the year from date or timestamp *x*.

All date and time functions use Universal Time (UT).

Example of a policy rules file

```
/*
Sample GPFS policy rules file
*/

rule 'vip' set pool 'pool0' where USER_ID <= 100
RULE 'm1' SET POOL 'pool1' WHERE LOWER(NAME) LIKE '%marc%'
RULE SET POOL 'pool1' REPLICATE (2) WHERE UPPER(NAME) = '%IBM%'
RULE 'r2' SET POOL 'pool2' WHERE UPPER(SUBSTRING(NAME FROM 1 FOR 4)) = 'GPFS'
RULE 'r3' SET POOL 'pool3' WHERE LOWER(SUBSTR(NAME,1,5)) = 'roger'
RULE SET POOL 'pool4' WHERE LENGTH(NAME) = 7
RULE SET POOL 'pool5' WHERE name like 'xyz%' AND name like '%qed' OR name like '%.tmp%'
RULE SET POOL 'pool6' WHERE name like 'abc%' OR name like '%xyz' AND name like 'x%'

RULE 'restore' RESTORE TO POOL 'pool0' where USER_ID <= 100


/* If none of the rules matches put those files in system pool */
rule 'default' SET POOL 'system'
```

Miscellaneous SQL functions:

The following miscellaneous SQL functions are available.

SNAP_ID(['*FilesetName*',] '*SnapshotName*')
Given an (optional) fileset/inode-space name and a snapshot name, this function returns the numeric snapshot ID of the given snapshot within the given inode-space.

GetEnv('*EnvironmentVariableName*')
This function gets the value of the specified environment variable.

GetMMconfig('*GPFSConfigurationParameter*')
This function gets the value of the specified GPFS configuration parameter.

The mmapplypolicy command and policy rules

The **mmapplypolicy** command has policy rules that are based on the characteristics of different phases.

Any given file is a potential candidate for at most one **MIGRATE** or **DELETE** operation during each invocation of the **mmapplypolicy** command. A single invocation of the **mmapplypolicy** command is called the *job*.

The **mmapplypolicy** command sets the SQL built-in variable **CURRENT_TIMESTAMP**, and collects pool occupancy statistics at the beginning of the job.

The **mmapplypolicy** job consists of three major phases:

1. "Phase one: Selecting candidate files"
2. "Phase two: Choosing and scheduling files" on page 40
3. "Phase three: Migrating and deleting files" on page 42

Phase one: Selecting candidate files

In the first phase of the **mmapplypolicy** job, all the files within the specified GPFS file system device, or below the input path name, are scanned. The attributes of each file are read from the file's GPFS inode structure.

Note: mmapplypolicy reads directly from the metadata disk blocks and can therefore lag behind the posix state of the file system. To be sure that `MODIFICATION_TIME` and the other timestamps are completely up to date, you can use the following suspend-and-resume sequence to force recent changes to disk:

```
mmfsctl fs-name suspend; mmfsctl fs-name resume;
```

For each file, the policy rules are considered, in order, from first rule to last:

- If the rule has a **WHEN** clause that evaluates to **FALSE**, the rule is skipped.
- If the rule has a **FROM POOL** clause, and the named pool does not match the **POOL_NAME** attribute of the file, the rule is skipped. A **FROM POOL** clause that specifies a group pool name matches a file if any pool name within the group pool matches the **POOL_NAME** attribute of the file.
- If there is a **THRESHOLD** clause and the current pool of the file has an occupancy percentage that is less than the *HighPercentage* parameter of the **THRESHOLD** clause, the rule is skipped.
- If the rule has a **FOR FILESET** clause, but none of the named filesets match the **FILESET_NAME** attribute of the file, the rule is skipped.
- If the rule has a **WHERE** clause that evaluates to **FALSE**, the rule is skipped. Otherwise, the rule applies.
- If the applicable rule is a **LIST** *'listname-y'* rule, the file becomes a candidate for inclusion in the named list unless the **EXCLUDE** keyword is present, in which case the file will not be a candidate; nor will any following **LIST** *'listname-y'* rules be considered for the subject file. However, the file is subject to **LIST** rules naming other list names.
- If the applicable rule is an **EXCLUDE** rule, the file will be neither migrated nor deleted. Files matching the **EXCLUDE** rule are not candidates for any **MIGRATE** or **DELETE** rule.

  Note: Specify the **EXCLUDE** rule before any other rules that might match the files that are being excluded. For example:

  ```
  RULE 'Exclude root's file' EXCLUDE where USER_ID = 0
  RULE 'Migrate all but root's files' MIGRATE TO POOL 'pool1'
  ```

  will migrate all the files that are not owned by **root**. If the **MIGRATE** rule was placed in the policy file before the **EXCLUDE** rule, all files would be migrated because the policy engine would evaluate the rules from first to last, and **root**'s files would have to match the **MIGRATE** rule.

  To exclude files from matching a **LIST** rule, you must create a separate **LIST** rule with the **EXCLUDE** clause and place it before the **LIST** rule.
- If the applicable rule is a **MIGRATE** rule, the file becomes a *candidate* for migration to the pool specified by the **TO POOL** clause.

  When a group pool is the **TO POOL** target of a **MIGRATE** rule, the selected files are distributed among the disk pools comprising the group pool, with files of highest weight going to the most preferred disk pool up to the occupancy limit for that pool. If there are still more files to be migrated, those go to the second most-preferred pool up to the occupancy limit for that pool (again choosing the highest-weight files from among the remaining selected files); and so on for the subsequent most-preferred pools, until either all selected files have been migrated or until all the disk pools of the group pool have been filled to their respective limits.

- If the applicable rule is a **DELETE** rule, the file becomes a *candidate* for deletion.
- If there is no applicable rule, the file is not a candidate for migration or deletion.
- Each candidate file (for migration or deletion) is also associated with a *LowPercentage* occupancy percentage value, which is taken from the **THRESHOLD** clause of the applicable rule. If not specified, the *LowPercentage* value defaults to 0%.
- Each candidate file is also associated with a numeric *weight*, either computed from the *WeightExpression* of the applicable rule, or assigned a default using these rules:
  - If a *LowPercentage* is specified within a **THRESHOLD** clause of the applicable rule, the weight of the candidate is taken as the **KB_ALLOCATED** attribute of the candidate file.
  - If a *LowPercentage* is not specified within a **THRESHOLD** clause of the applicable rule, the weight of the candidate is taken as **+infinity**.

Phase two: Choosing and scheduling files

In the second phase of the **mmapplypolicy** job, some or all of the candidate files are chosen.

Chosen files are scheduled for migration or deletion, taking into account the weights and thresholds determined in "Phase one: Selecting candidate files" on page 39, as well as the actual pool occupancy percentages. Generally, candidates with higher weights are chosen ahead of those with lower weights.

File migrations to and from external pools are done before migrations and deletions that involve only GPFS disk pools.

File migrations that do not target group pools are done before file migrations to group pools.

File migrations that target a group pool are done so that candidate files with higher weights are migrated to the more preferred GPFS disk pools within the group pool, but respecting the **LIMIT**s specified in the group pool definition.

The following two options can be used to adjust the method by which candidates are chosen:

--choice-algorithm {best | exact | fast}
Specifies one of the following types of algorithms that the policy engine is to use when selecting candidate files:

best
Chooses the optimal method based on the rest of the input parameters.

exact
Sorts all of the candidate files completely by weight, then serially considers each file from highest weight to lowest weight, choosing feasible candidates for migration, deletion, or listing according to any applicable rule **LIMIT**s and current storage-pool occupancy. This is the default.

fast
Works together with the parallelized **-g /shared-tmp -N** node-list selection method. The **fast** choice method does not completely sort the candidates by weight. It uses a combination of statistical, heuristic, and parallel computing methods to favor higher weight candidate files over those of lower weight, but the set of chosen candidates may be somewhat different than those of the **exact** method, and the order in which the candidates are migrated, deleted, or listed is somewhat more random. The **fast** method uses statistics gathered during the policy evaluation phase. The **fast** choice method is especially fast when the collected statistics indicate that either all or none of the candidates are feasible.

--split-margin *n.n*
A floating-point number that specifies the percentage within which the **fast**-choice algorithm is

allowed to deviate from the **LIMIT** and **THRESHOLD** targets specified by the policy rules. For example if you specified a **THRESHOLD** number of 80% and a split-margin value of 0.2, the **fast**-choice algorithm could finish choosing files when it reached 80.2%, or it might choose files that bring the occupancy down to 79.8%. A nonzero value for split-margin can greatly accelerate the execution of the **fast**-choice algorithm when there are many small files. The default is 0.2.

File grouping and the SIZE clause

When scheduling files, **mmapplypolicy** simply groups together either the next 100 files by default, or the number of files explicitly set using the **-B** option.

However, you can set up **mmapplypolicy** to schedule files so that each invocation of the InterfaceScript gets approximately the same amount of file data to process. To do so, use the **SIZE** clause of certain policy rules to specify that scheduling be based on the sum of the sizes of the files. The **SIZE** clause can be applied to the following rules (for details, see "Policy rules" on page 21):
- **DELETE**
- **EXTERNAL LIST**
- **EXTERNAL POOL**
- **LIST**
- **MIGRATE**

Administrator-specified customized file grouping or aggregation

In addition to using the **SIZE** clause to control the *amount* of work passed to each invocation of a InterfaceScript, you can also specify that files with *similar attributes* be grouped or aggregated together during the scheduling phase. To do so, use an aggregator program to take a list of chosen candidate files, sort them according to certain attributes, and produce a reordered file list that can be passed as input to the user script.

You can accomplish this by following these steps:
1. Run **mmapplypolicy** with the **-I prepare** option to produce a list of chosen candidate files, but not pass the list to a InterfaceScript.
2. Use your aggregator program to sort the list of chosen candidate files into groups with similar attributes and write each group to a new, separate file list.
3. Run **mmapplypolicy** with the **-r** option, specifying a set of file list files to be read. When invoked with the **-r** option, **mmapplypolicy** does not choose candidate files; rather, it passes the specified file lists as input to the InterfaceScript.

   Note: You can also use the **-q** option to specify that small groups of files are to be taken in round-robin fashion from the input file lists (for example, take a small group of files from x.list.A, then from x.list.B, then from x.list.C, then back to x.list.A, and so on, until all of the files have been processed).

   To prevent **mmapplypolicy** from redistributing the grouped files according to size, omit the **SIZE** clause from the appropriate policy rules and set the bunching parameter of the **-B** option to a very large value.

Reasons for candidates not to be chosen for deletion or migration

Generally, a candidate is not chosen for deletion from a pool, nor migration out of a pool, when the pool occupancy percentage falls below the *LowPercentage* value. Also, candidate files will not be chosen for migration into a target **TO POOL** when the target pool reaches the occupancy percentage specified by the **LIMIT** clause (or 99% if no **LIMIT** was explicitly specified by the applicable rule).

The limit clause does not apply when the target **TO POOL** is a group pool; the limits specified in the rule defining the target group pool govern the action of the **MIGRATE** rule. The policy-interpreting program (for example, **mmapplypolicy**) may issue a warning if a **LIMIT** clause appears in a rule whose target pool is a group pool.

Phase three: Migrating and deleting files

In the third phase of the **mmapplypolicy** job, the candidate files that were chosen and scheduled by the second phase are migrated or deleted, each according to its applicable rule.

For migrations, if the applicable rule had a **REPLICATE** clause, the replication factors are also adjusted accordingly. It is acceptable for the effective **FROM POOL** and **TO POOL** to be the same because the **mmapplypolicy** command can be used to adjust the replication factors of files without necessarily moving them from one pool to another.

The migration performed in the third phase can involve large amounts of data movement. Therefore, you may want to consider using the **–I defer** option of the **mmapplypolicy** command, and then perform the data movements with the **mmrestripefs -p** command.

Policy rules: Examples and tips

Before you write and apply policies, consider the following advice.

You are advised to test your rules using the **mmapplypolicy** command with the **-I test** option and the **-L 3** (or higher) option. This will help you understand which files are selected as candidates, and which candidates are chosen.

Do not apply a policy to an entire file system of vital files until you are confident that the rules correctly express your intentions. To test your rules, find or create a subdirectory with a modest number of files, some that you expect to be selected by your SQL policy rules and some that you expect will be skipped.

Then run the following command:

```
mmapplypolicy /TestSubdirectory  -L 6  -I test
```

The output will show you exactly which files are scanned, and which match rules or no rules. If a problem is not apparent, you can add a SHOW() clause to your rule or rules to examine the values of the file attributes of interest or the value of any SQL expression. To examine several, use the following:

```
SHOW('x1=' || varchar(Expression1) || ' x2=' || varchar(Expression2) || ... )
```

where *ExpressionX* is the SQL variable or expression of function that you suspect or do not understand. Beware that if any expression evaluates to SQL NULL, the entire show clause will be NULL, by the rules of SQL. One way to show null vs. non-null values is to define a macro and use it as in the following example:

```
define(DISPLAY_NULL,[CASE WHEN ($1) IS NULL THEN '_NULL_' ELSE varchar($1) END])

rule list a SHOW( 'x1=' || DISPLAY_NULL(xattr('user.marc')) || ' and  x2=' || DISPLAY_NULL(xattr('user.eric')))
```

Note: For examples and more information on the **-L** flag, see the topic *The mmapplypolicy -L command* in the *IBM Spectrum Scale: Problem Determination Guide.*

1. Delete files from the storage pool named **pool_1** that have not been accessed in the last 30 days, and are named like temporary files or appear in any directory that is named **tmp**:

```
RULE 'del1' DELETE FROM POOL 'pool_1'
   WHERE (DAYS(CURRENT_TIMESTAMP) – DAYS(ACCESS_TIME) > 30)
        AND (lower(NAME) LIKE '%.tmp' OR PATH_NAME LIKE '%/tmp/%')
```

2. Use the SQL **LIKE** predicate to test file names and path names:

```
RULE '*/_*'  DELETE WHERE PATH_NAME LIKE '%/x_%' ESCAPE 'x'
RULE '*XYZ*' DELETE WHERE NAME LIKE '%XYZ%'
RULE '12_45' DELETE WHERE NAME LIKE '12x_45' ESCAPE 'x'
RULE '12%45' DELETE WHERE NAME LIKE '12x%45' ESCAPE 'x'
RULE '12?45' DELETE WHERE NAME LIKE '12_45'
RULE '12*45' DELETE WHERE NAME LIKE '12%45'
RULE '*_*'   DELETE WHERE NAME LIKE '%x_%' ESCAPE 'x'
```

Where:

- A percent % wildcard in the name represents zero or more characters.
- An underscore (_) wildcard in the name represents one byte.

Use the optional **ESCAPE** clause to establish an escape character, when you need to match '_' or '%' exactly.

3. Use the SQL **UPPER** and **LOWER** functions to ignore case when testing names:

```
RULE 'UPPER'  DELETE WHERE upper(PATH_NAME) LIKE '%/TMP/OLD/%'
RULE 'lower'  DELETE WHERE lower(PATH_NAME) LIKE '%/tmp/old/%'
```

4. Use the SQL **SUBSTR** or **SUBSTRING** functions to test a substring of a name:

```
RULE 's1' DELETE WHERE SUBSTRING(NAME FROM 1 FOR 5)='XXXX-'
RULE 's2' DELETE WHERE SUBSTR(NAME,1,5)='YYYY-'
```

5. Use the SQL **SUBSTR** and **LENGTH** functions to test the suffix of a name:

```
RULE 'sfx' DELETE WHERE SUBSTR(NAME,LENGTH(NAME)-3)='.tmp'
```

6. Use a **WHEN** clause to restrict rule applicability to a particular day of the week:

```
RULE 'D_SUN' WHEN (DayOfWeek(CURRENT_DATE)=1) /* Sunday */
  DELETE WHERE PATH_NAME LIKE '%/tmp/%'
```

CURRENT_DATE is an SQL built in operand that returns the date portion of the **CURRENT_TIMESTAMP** value.

7. Use the SQL **IN** operator to test several possibilities:

```
RULE 'D_WEEKEND' WHEN (DayOfWeek(CURRENT_DATE) IN (7,1)) /*  Saturday or Sunday */
  DELETE WHERE PATH_NAME LIKE '%/tmp/%'
```

For information on how to use a macro processor such as **m4** to make reading and writing policy rules easier, see "Using macro processing utilities with policy rules" on page 45.

8. Use a **FILESET** clause to restrict the rule to files within particular filesets:

```
RULE 'fsrule1' MIGRATE TO POOL 'pool_2'
        FOR FILESET('root','fset1')
```

In this example there is no **FROM POOL** clause, so regardless of their current storage pool placement, all files from the named filesets are subject to migration to storage pool **pool_2**.

9. Use an **EXCLUDE** rule to exclude a set of files from all subsequent rules:

```
RULE 'Xsuper' EXCLUDE WHERE USER_ID=0
RULE 'mpg'    DELETE  WHERE lower(NAME) LIKE '%.mpg' AND FILE_SIZE>20123456
```

Notes:

a. Specify the **EXCLUDE** rule before rules that might match the files that are being excluded.

b. You cannot define a list and what to exclude from the list in a single rule. You must define two LIST statements, one specifying which files will be in the list, and one specifying what to exclude from the list. For example, to exclude files containing the word **test** from the LIST rule **allfiles**, define the following:

```
RULE EXTERNAL LIST 'allfiles' EXEC '/u/brownap/policy/CHE/exec.list'

RULE 'exclude_allfiles' LIST 'allfiles' EXCLUDE where name like '%test%'

RULE 'all' LIST 'allfiles' SHOW('misc_attr ='|| MISC_ATTRIBUTES || HEX(MISC_ATTRIBUTES)) \
  where name like '%'
```

10. Use the SQL **NOT** operator with keywords, along with **AND** and **OR**:

```
RULE 'D_WEEKDAY' WHEN (DayOfWeek(CURRENT_DATE) NOT IN (7,1)) /*  a weekday */
  DELETE WHERE (PATH_NAME LIKE '%/tmp/%' OR NAME LIKE '%.tmp')
          AND (KB_ALLOCATED > 9999 AND NOT USER_ID=0)
```

11. Use a **REPLICATE** clause to increase the availability of selected files:

    ```
    RULE 'R2' MIGRATE FROM POOL 'ypooly' TO POOL 'ypooly'
      REPLICATE(2) WHERE USER_ID=0
    ```

    Before increasing the data replication factor for any file, the file system must be configured to support data replication.

12. The difference of two SQL Timestamp values may be compared to an SQL Interval value:

    ```
    rule 'a' migrate to pool 'A' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' DAYS
    rule 'b' migrate to pool 'B' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' HOURS
    rule 'c' migrate to pool 'C' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' MINUTES
    rule 'd' migrate to pool 'D' where CURRENT_TIMESTAMP - MODIFICATION_TIME > INTERVAL  '10' SECONDS
    ```

    For the best precision, use the `INTERVAL...SECONDS` construct.

13. By carefully assigning both weights and thresholds, the administrator can formally express rules like this:

    If the storage pool named **pool_X** has an occupancy percentage above 90% now, bring the occupancy percentage of storage pool named **pool_X** down to 80% by migrating files that are three months or older to the storage pool named **pool_ZZ**. But, if you can find enough year-old files to bring the occupancy percentage down to 50%, do that also.

    ```
    RULE 'year-old' MIGRATE FROM POOL 'pool_X'
       THRESHOLD(90,50) WEIGHT(weight_expression)
       TO POOL 'pool_ZZ'
       WHERE  DAYS(CURRENT_TIMESTAMP) – DAYS(ACCESS_TIME) > 365

    RULE '3month-old' MIGRATE FROM POOL 'pool_X'
       THRESHOLD(90,80) WEIGHT(weight_expression)
       TO POOL 'pool_ZZ'
       WHERE  DAYS(CURRENT_TIMESTAMP) – DAYS(ACCESS_TIME) > 90
    ```

    More information about weights is available in the next example.

    A goal of this **mmapplypolicy** job is to reduce the occupancy percentage of the **FROM POOL** to the low occupancy percentage specified on the **THRESHOLD** clause, if possible. The **mmapplypolicy** job does not migrate or delete more files than are necessary to produce this occupancy percentage. The task consists of these steps:

    a. Each candidate file is assigned a weight.
    b. All candidate files are sorted by weight.
    c. The highest weight files are chosen to **MIGRATE** or **DELETE** until the low occupancy percentage is achieved, or there are no more candidates.

    The administrator who writes the rules must ensure that the computed weights are as intended, and that the comparisons are meaningful. This is similar to the TSM convention, where the weighting function for each file is determined by the equation:

    ```
    X * access_age + Y * file_size
    ```

    where:

    access_age is **DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)**

    file_size is **FILE_SIZE** or **KB_ALLOCATED**

    X and Y are weight factors chosen by the system administrator.

14. The **WEIGHT** clause can be used to express ideas like this (stated informally):

    ```
    IF access_age > 365 days THEN weight = 100000 + access_age
    ELSE IF access_age < 30 days THEN weight = 0
    ELSE weight= KB_ALLOCATED
    ```

    This means:

    - Give a very large weight bias to any file older than a year.

- Force the weight of any file younger than 30 days to 0.
- Assign weights to all other files according to the number of kilobytes occupied.

The formal SQL syntax is this:

```
CASE
 WHEN DAYS(CURRENT_TIMESTAMP) – DAYS(ACCESS_TIME) > 365
  THEN 100000 + DAYS(CURRENT_TIMESTAMP) – DAYS(ACCESS_TIME)
 WHEN DAYS(CURRENT_TIMESTAMP) – DAYS(ACCESS_TIME) < 30
  THEN 0
 ELSE
  KB_ALLOCATED
END
```

15. The **SHOW** clause has no effect in matching files but can be used to define additional attributes to be exported with the candidate file lists. It may be used for any purpose but is primarily used to support file aggregation.

    To support aggregation, you can use the **SHOW** clause to output an aggregation value for each file selected by a rule. You can then output those values to a file list and input that list to an external program that groups the files into aggregates.

16. If you have a large number of filesets against which to test, use the **FILESET_NAME** variable as shown in the following example:

    ```
    RULE 'x' SET POOL 'gold' WHERE FILESET_NAME LIKE 'xyz.%.xyz'
    ```

    However, if you are testing against just a few filesets, you can use the `FOR FILESET('xyz1', 'xyz2')` form instead.

17. You may convert a time interval value to a number of seconds using SQL cast syntax; for example:

    ```
    define([toSeconds],[(($1) SECONDS(12,6))])

    define([toUnixSeconds],[toSeconds($1 - '1970-1-1@0:00')])

    RULE external list b
    RULE list b SHOW('sinceNow=' toSeconds(current_timestamp-modification_time) )
    RULE external list c
    RULE list c SHOW('sinceUnixEpoch=' toUnixSeconds(modification_time) )
    ```

    The following is also supported:

```
define(access_age_in_days,( INTEGER(( (CURRENT_TIMESTAMP - ACCESS_TIME) SECONDS)) /(24*3600.0) ) )

RULE external list w exec ''
RULE list w weight(access_age_in_days) show(access_age_in_days)
```

Using macro processing utilities with policy rules

Prior to evaluating the policy rules, GPFS invokes the **m4** macro processor to process the policy file.

This processing allows you to incorporate into the policy file some of the traditional **m4** facilities and to define simple and parameterized macros, conditionally include text, perform conditional evaluation, perform simple string operations, perform simple integer arithmetic and much more.

Note: GPFS uses the **m4** built-in **changequote** macro to change the quote pair to [ ] and the **changecom** macro to change the comment pair to /* */ (as in the C programming language).

Utilizing **m4** as a front-end processor simplifies writing policies and produces policies that are easier to understand and maintain. Here is Example 14 on page 44 from "Policy rules: Examples and tips" on page 42 written with a few **m4** style macro definitions:

```
define(access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)))

define(weight_expression,
       CASE
        WHEN access_age > 365
          THEN 100000 + access_age
```

```
        WHEN access_age < 30
          THEN 0
        ELSE
          KB_ALLOCATED
        END
)

RULE year-old MIGRATE FROM POOL pool_X
   THRESHOLD(90,50) WEIGHT(weight_expression)
   TO POOL pool_ZZ
   WHERE access_age > 365

RULE 3month-old MIGRATE FROM POOL pool_X
   THRESHOLD(90,80) WEIGHT(weight_expression)
   TO POOL pool_ZZ
   WHERE access_age > 90
```

If you would like to use megabytes or gigabytes instead of kilobytes to represent file sizes, and **SUNDAY**, **MONDAY**, and so forth instead of 1, 2, and so forth to represent the days of the week, you can use macros and rules like this:

```
define(MB_ALLOCATED,(KB_ALLOCATED/1024.0))
define(GB_ALLOCATED,(KB_ALLOCATED/1048576.0))
define(SATURDAY,7)
define(SUNDAY,1)
define(MONDAY,2)
define(DAY_OF_WEEK, DayOfWeek(CURRENT_DATE))

RULE 'gb1' WHEN(DAY_OF_WEEK IN (SATURDAY,SUNDAY))
        MIGRATE TO POOL 'ypooly' WHERE GB_ALLOCATED >= .015

RULE 'mb4' MIGRATE TO POOL 'zpoolz' WHERE MB_ALLOCATED >= 4
```

The **mmapplypolicy** command provides a **-M** option that can be used to specify **m4** macro definitions when the command is invoked. The policy rules may include variable identifiers whose values can be set using one or more **-M** options on the **mmapplypolicy** command. The policy rules could then compare file attributes to the currently provided values for the macro defined variables.

Among other things, this allows you to create a single policy file and reuse it for incremental backups without editing the file for each backup. For example, if your policy file contains the rules:

```
RULE EXTERNAL POOL 'archive' EXEC '/opts/hpss/archiveScript' OPTS '-server archive_server'
RULE 'mig1' MIGRATE TO POOL 'dead' WHERE ACCESS_TIME < TIMESTAMP(deadline)
RULE 'bak1' MIGRATE TO POOL 'archive' WHERE MODIFICATION_SNAPID > last_snapid
```

Then, if you invoke **mmapplypolicy** with these options

```
mmapplypolicy ... -M "deadline='2006-11-30'" -M "last_snapid=SNAPID('2006_DEC')" \
-M archive_server="archive.abc.com"
```

The "`mig1`" rule will migrate old files that were not accessed since 2006/11/30 to an online pool named "dead". The "`bak1`" rule will migrate files that have changed since the 2006_DEC snapshot to an external pool named "`archive`". When the external script `/opts/hpss/archiveScript` is invoked, its arguments will include "`-server archive.abc.com`".

Managing policies

Policies and the rules that they contain are used to assign files to specific storage pools.

A storage pool typically contains a set of volumes that provide a specific quality of service for a specific use, such as to store all files for a particular application or a specific business division.

Managing policies includes:

- "Creating a policy"
- "Installing a policy"
- "Changing the active policy"
- "Listing policies" on page 48
- "Validating policies" on page 48
- "Deleting policies" on page 48

Creating a policy

Create a text file for your policy by following these guidelines.

- A policy must contain at least one rule.
- A policy file is limited to a size of 1 MB.
- The last placement rule of a policy rule list should be of this form, so that if no other placement rules apply to a file, the file will be assigned to a default pool:

  ```
  RULE 'DEFAULT' SET POOL 'default-data-pool'
  ```

  If you do not do this, and no other rule applies, an **EINVAL** error code is returned.

  For file systems that are upgraded to V4.1.1 or later: If there are no **SET POOL** policy rules installed to a file system by **mmchpolicy**, the system acts as if the single rule **SET POOL** '*first-data-pool*' is in effect, where *first-data-pool* is the firstmost non-system pool that is available for file data storage, if such a non-system pool is available. ("Firstmost" is the first according to an internal index of all pools.) However, if there are no policy rules installed and there is no non-system pool, the system acts as if **SET POOL 'system'** is in effect.

  For file systems that are upgraded to V4.1.1: Until a file system is upgraded, if no **SET POOL** rules are present (set by **mmchpolicy**) for the file system, all data is stored in the **'system'** pool.
- Comments within a policy must start with a **/*** and end with a ***/**:

  ```
  /* This is a comment */
  ```

See "Policy rules" on page 21

Installing a policy

Install a policy by following these guidelines.

To install a policy:

1. Create a text file containing the desired policy rules.
2. Issue the **mmchpolicy** command.

Changing the active policy

When you prepare a file with the new or changed policy rules, then issue the **mmchpolicy** command.

The **mmchpolicy** command activates the following sequence of events:

1. The policy file is read into memory, and the information is passed to the current file system manager node.
2. The policy rules are validated by the file system manager.
3. If the policy file contains incorrect rules, no updates are made and an error is returned.
4. If no errors are detected, the new policy rules are installed in an internal file.

Policy changes take effect immediately on all nodes that have the affected file system mounted. For nodes that do not have the file system mounted, policy changes take effect upon the next mount of the file system.

Listing policies

When you use the **mmlspolicy** command to list policies, follow these guidelines.

The **mmlspolicy** command displays policy information for a given file system. The information displayed is:

- When the policy file was installed.
- The user who installed the policy file.
- The first line of the original policy file.

The **mmlspolicy -L** command returns the installed (original) policy file. This shows all the rules and comments as they were in the policy file when it was installed. This is useful if you want to change policy rules - simply retrieve the original policy file using the **mmlspolicy -L** command and edit it.

Validating policies

When you validate a policy file, follow this guideline.

The **mmchpolicy -I test** command validates but does *not* install a policy file.

Deleting policies

When you remove the current policy rules and restore the file-placement policy, follow this guideline.

To remove the current policy rules and restore the default GPFS file-placement policy, specify **DEFAULT** as the name of the policy file on the **mmchpolicy** command. This is equivalent to installing a policy file with just one rule:

```
RULE 'DEFAULT' SET POOL 'system'
```

Improving performance with the --sort-command parameter

To improve performance of the **mmapplypolicy** command, follow these guidelines.

One possible way to improve the performance of the **mmapplypolicy** command is to specify an alternative sort command to be used instead of the default sort command provided by the operating system. To do this, issue **mmapplypolicy --sort-command** *SortCommand*, specifying the executable path of the alternative command.

For example, on AIX the GNU **sort** program, freely available within the `coreutils` package from AIX Toolbox for Linux Applications (www.ibm.com/systems/power/software/aix/linux/toolbox), will typically perform large sorting tasks much faster than the standard AIX sort command. If you wanted to specify the GNU sort program, you would use the following command: `mmapplypolicy --sort-command /opt/freeware/bin/sort`.

Before issuing **mmapplypolicy --sort-command** on a large number of files, first do a performance comparison between the alternative sort command and the default sort command on a smaller number of files to determine whether the alternative command is in fact faster.

If you specify an alternative sort command, it is recommended that you install it on all cluster nodes.

Working with external storage pools

With external storage pools you can migrate files to storage pools managed by an external application such as IBM Spectrum Protect.

The following topics describe how to work with external storage pools:

- Defining the external pools
- "User-provided program for managing external pools" on page 49
- "File list format" on page 50

- "Record format" on page 50
- "Migrate and recall with external pools" on page 52
- "Pre-migrating files with external storage pools" on page 52
- "Purging files from external storage pools" on page 53
- "Using thresholds with external pools" on page 53

Defining external pools

When you define external pools, follow these rules.

GPFS file management policy rules control data migration into external storage pools. Before you can write a migration policy you must define the external storage pool that the policy will reference. After you define the storage pool, you can then create policies that set thresholds that trigger data migration into or out of the referenced external pool.

When a storage pool reaches the defined threshold or when you invoke **mmapplypolicy**, GPFS processes the metadata, generates a list of files, and invokes a user provided script or program which initiates the appropriate commands for the external data management application to process the files. This allows GPFS to transparently control offline storage and provide a tiered storage solution that includes tape or other media.

Before you can migrate data to an external storage pool, you must define that pool. To define external storage pools, use a GPFS policy rule as follows:

```
RULE EXTERNAL POOL 'PoolName' EXEC 'InterfaceScript' [OPTS 'OptionsString'] [ESCAPE 'SpecialCharacters']
```

Where:
- *PoolName* defines the name of the storage pool
- *InterfaceScript* defines the program or script to be invoked to migrate data to or from the external pool
- *OptionsString* is an optional string that, if provided, will be passed to the *InterfaceScript*

You must have a separate EXTERNAL POOL rule for each external pool that you wish to define.

Example of a rule that defines a storage pool

The following rule defines a storage pool called `externalpoolA`.

```
RULE EXTERNAL POOL 'externalpoolA' EXEC '/usr/hsm/bin/hsmControl' OPTS '-server=hsm-manager.nyc.com'
```

In this example:
- `externalpoolA` is the name of the external pool
- `/usr/hsm/bin/hsmControl` is the location of the executable script that will be invoked when there are files for migration
- `-server=hsm-manager.nyc.com` is the location of storage pool `externalpoolA`

For additional information, refer to "User-provided program for managing external pools."

User-provided program for managing external pools

After you define an external storage pool, subsequent migration or deletion rules might refer to that pool as a source or target storage pool.

When the **mmapplypolicy** command is invoked and a rule dictates that data should be moved to or from an external pool, the user provided program identified with the **EXEC** clause in the policy rule launches. That executable program receives three arguments:
- The command to be executed. Your script should implement each of the following sub-commands:
  - LIST - Provides arbitrary lists of files with no semantics on the operation.

  - MIGRATE - Migrate files to external storage and reclaim the online space allocated to the file.
  - PREMIGRATE - Migrate files to external storage but do not reclaim the online space.
  - PURGE - Delete files from both the online file system and the external storage.
  - RECALL - Recall files from external storage to the online storage.
  - TEST – Test for presence and operation readiness. Return zero for success. Return non-zero if the script should not be used on a given node.
- The name of a file containing a list of files to be migrated, premigrated, or purged. See "File list format" for detailed description of the layout of the file.
- Any optional parameters specified with the OPTS clause in the rule. These optional parameters are not interpreted by the GPFS policy engine.

The **mmapplypolicy** command invokes the external pool script on all nodes in the cluster that have installed the script in its designated location. The script must be installed at the node that runs **mmapplypolicy**. You can also install the script at other nodes for parallel operation but that is not required. GPFS may call your exit script one or more times for each command.

Important: Use the **EXCLUDE** rule to exclude any special files that are created by an external application. For example, when using Tivoli Storage Manager (TSM) or Hierarchical Storage Management (HSM), exclude the **.SpaceMan** directory to avoid migration of **.SpaceMan**, which is an HSM repository.

File list format

Each call to the external pool script specifies the pathname for a temporary file that contains a list of files to be operated on.

This file list defines one file per line as follows:

```
InodeNumber GenNumber SnapId [OptionalShowArgs] -- FullPathToFile
```

where:

- *InodeNumber* is a 64-bit inode number.
- *GenNumber* is a 32-bit file generation number.
- *SnapId* is a 64-bit snapshot identifier.
- *OptionalShowArgs* is the result, if any, from the evaluation of the SHOW clause in the policy rule.
- *FullPathToFile* is a fully qualified path name to the file. When there are multiple paths within a file system to a particular file (*Inode*, *GenNumber*, and *SnapId*), each path is shown.
- The "--" characters are a field delimiter that separates the optional show parameters from the path name to the file.

Note: GPFS does not restrict the character set used for path and file names. All characters except '`\0`' are valid. To make the files readily parsable, files or directories containing the newline character and/or other special characters are "escaped", as described previously, in connection with the `ESCAPE '%special-characters'` clause.

Record format

The format of the records in each file list file can be expressed as shown in the following example.

Each file list file:

```
iAggregate:WEIGHT:INODE:GENERATION:SIZE:iRule:resourceID:attr_flags:
path-length!PATH_NAME:pool-length!POOL_NAME
[;show-length>!SHOW]end-of-record-character
```

where:

- *iAggregate* is a grouping index that is assigned by **mmapplypolicy**.
- *WEIGHT* represents the **WEIGHT** policy language file attribute.

- *INODE* represents the **INODE** policy language file attribute.
- *GENERATION* represents the **GENERATION** policy language file attribute.
- *SIZE* represents the **SIZE** policy language file attribute.
- *iRule* is a rule index number assigned by **mmapplypolicy**, which relates to the policy rules file that is supplied with the **-P** argument.
- *resourceID* represents a pool index, **USER_ID**, **GROUP_ID**, or fileset identifier, depending on whether thresholding is done with respect to pool usage or to user, group, or fileset quotas.
- *attr_flags* represents a hexadecimal encoding of some of the attributes that are also encoded by the policy language variable *MISC_ATTRIBUTES*. The low-order 20 bits of *attr_flags* are taken from the **ia_flags** word that is defined in the **gpfs.h** API definition.
- *path-length* represents the length of the character string PATH_NAME.
- *pool-length* represents the length of the character string POOL_NAME.
- *show-length* represents the length of the character string SHOW.
- *end-of-record-character* is **\n** or **\0**.

Note: You can only change the values of the *iAggregate*, *WEIGHT*, *SIZE*, and *attr_flags* fields. Changing the values of other fields can cause unpredictable policy execution results.

All of the numeric fields are represented as hexadecimal strings, except the *path-length*, *pool-length*, and *show-length* fields, which are decimal encoded. These fields can be preceded by a minus sign (**-**), which indicates that the string that follows it contains escape sequences. In this case, the string might contain occurrences of the character pair **\n**, which represents a single newline character with a hexadecimal value of 0xA in the filename. Also, the string might contain occurrences of the character pair \\, which represents a single \ character in the filename. A \ will only be represented by \\ if there are also newline characters in the filename. The value of the length field within the record counts any escape characters.

The encoding of *WEIGHT* is based on the 64-bit IEEE floating format, but its bits are *flipped* so that when a file list is sorted using a conventional collating sequence, the files appear in decreasing order, according to their *WEIGHT*.

The encoding of *WEIGHT* can be expressed and printed using C++ as:

```
double w =  - WEIGHT;
/* This code works correctly on big-endian and little-endian systems */
uint64 u = *(uint64*)&w;  /* u is a 64 bit long unsigned integer
     containing the IEEE 64 bit encoding of the double floating point
     value of variable w */
     uint64 hibit64 = ((uint64)1<<63);
if (w < 0.0) u = ~u;  /* flip all bits */
else u = u | hibit64; /* force the high bit from 0 to 1,
      also handles both "negatively" and "positively" signed 0.0 */
printf("%016llx",u);
```

The format of the majority of each record can be expressed in C++ as:

```
printf("%03x:%016llx:%016llx:%llx:%llx:%x:%x:%llx:%d!%s:%d!%s",
  iAggregate, u /*encoding of –1*WEIGHT from above*/, INODE, ... );
```

Notice that the first three fields are fixed in length to facilitate the sorting of the records by the field values *iAggregate*, *WEIGHT*, and *INODE*.

The format of the optional SHOW string portion of the record can be expressed as:

```
if(SHOW && SHOW[0]) printf(";%d!%s",strlen(SHOW),SHOW);
```

For more information, see the topic *mmapplypolicy command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Migrate and recall with external pools

After you define an external storage pool, subsequent migration or deletion rules might refer to that pool as a source or target storage pool.

When you invoke **mmapplypolicy** and a rule dictates that data should be deleted or moved to or from an external pool, the program identified in the EXTERNAL POOL rule is invoked with the following arguments:

- The command to be executed.
- The name of the file containing a list of files to be migrated, pre-migrated, or purged.
- Optional parameters, if any.

For example, let us assume an external pool definition:

```
RULE EXTERNAL POOL 'externalpoolA'
   EXEC '/usr/hsm/bin/hsmControl' OPTS '-server=hsm-manager.nyc.com'
```

To move files from the internal **system** pool to storage pool "externalpoolA" you would simply define a migration rule that may look something like this:

```
RULE 'MigToExt' MIGRATE FROM POOL('system') TO POOL('externalpoolA') WHERE ...
```

This would result in the external pool script being invoked as follows:

```
/usr/hsm/bin/hsmControl MIGRATE /tmp/filelist -server=hsm-manager.nyc.com
```

Similarly, a rule to migrate data from an external pool back to an internal storage pool could look like:

```
RULE 'MigFromExt' MIGRATE FROM POOL 'externalpoolA' TO POOL 'system' WHERE ...
```

This would result in the external pool script being invoked as follows:

```
/usr/hsm/bin/hsmControl RECALL /tmp/filelist -server=hsm-manager.nyc.com
```

Notes:

1. When migrating to an external storage pool, GPFS ignores the LIMIT and REPLICATION clauses in the policy rule.
2. If you are using HSM with external storage pools, you may need to create specific rules to avoid system problems. These rules should exclude HSM-related system files from both migration and deletion. These rules use the form:
   ```
   RULE 'exclude hsm system files' EXCLUDE WHERE PATH_NAME LIKE '%/.SpaceMan%'
   ```

Pre-migrating files with external storage pools

Pre-migration is a standard technique of Hierarchical Storage Management (HSM) systems such as Tivoli Storage Manager.

Pre-migration copies data from GPFS internal storage pools to external pools but leaves the original data online in the active file system. Pre-migrated files are often referred to as "dual resident" to indicate that the data for the files are available both online in GPFS and offline in the external storage manager. Files in the pre-migrated state allow the external storage manager to respond more quickly to low space conditions by simply deleting the copy of the file data that is stored online.

The files to be pre-migrated are determined by the policy rules that migrate data to an external storage pool. The rule will select files to be migrated and optionally select additional files to be pre-migrated. The THRESHOLD clause of the rule determines the files that need to be pre-migrated.

If you specify the THRESHOLD clause in file migration rules, the **mmapplypolicy** command selects files for migration when the affected storage pool reaches the specified high occupancy percentage threshold. Files are migrated until the storage pool utilization is reduced to the specified low occupancy percentage threshold. When migrating to an external storage pool, GPFS allows you to specify a third pool occupancy percentage which defines the file pre-migration threshold: after the low occupancy percentage is reached, files are pre-migrated until the pre-migration occupancy percentage is reached.

To explain thresholds in another way, think of an internal storage pool with a high threshold of 90%, a low threshold of 80%, and a pre-migrate threshold of 60%. When this internal storage pool reaches 90% occupancy, the policy rule will migrate files until the occupancy of the pool reaches 80% then it will continue to pre-migrate another 20% of the file space until the 60% threshold is reached.

Pre-migration can only be done with external storage managers using the XDSM Data Storage Management API (DMAPI). Files in the migrated and pre-migrated state will have a DMAPI managed region set on the file data. Files with a managed region are visible to **mmapplypolicy** and may be referenced by a policy rule. You can approximate the amount of pre-migrated space required by counting the space used after the end of the first full data block on all files with managed regions.

Note:

1. If you do not set a pre-migrate threshold or if you set a value that is greater than or equal to the low threshold, then GPFS will not pre-migrate files. This is the default setting.
2. If you set the pre-migrate threshold to zero, then GPFS will pre-migrate all files.

Purging files from external storage pools

Files that have been migrated to an external storage pool continue to have their file name and attributes stored in GPFS; only the file data has been migrated. Files that have been migrated or pre-migrated to an external storage pool may be deleted from the GPFS internal storage pool and from the external storage pool with the policy language using a DELETE rule.

```
RULE 'DelFromExt' DELETE WHERE ...
```

If the file has been migrated or pre-migrated, this would result in the external pool script being invoked as follows:

```
/usr/hsm/bin/hsmControl PURGE /tmp/filelist -server=hsm-manager.nyc.com
```

The script should delete a file from both the online file system and the external storage manager. However, most HSM systems automatically delete a file from the external storage manager whenever the online file is deleted. If that is how your HSM system functions, your script will only have to delete the online file.

Using thresholds with external pools

Exhausting space in any one online storage pool generates a NO_SPACE event even though there might be space available in other online storage pools. To create free space, file data can be moved to other online storage pools, deleted, or moved to external storage pools.

Under most conditions, Hierarchical Storage Management (HSM) systems try to avoid NO_SPACE events by monitoring file system space usage and migrating data to near-line storage when the system exceeds a specified threshold. You can set up a policy to monitor space usage using a threshold. When GPFS
| reaches a similar threshold, it generates a **lowDiskSpace** event, which triggers **mmapplypolicy** to generate a list of files to be migrated to external storage.

| A NO_SPACE event is generated if the file system is out of space. A **lowDiskSpace** event requires a
| threshold. If a threshold is specified, a **lowDiskSpace** event is generated.

| GPFS provides user exits for NO_SPACE and **lowDiskSpace** events. Using the **mmaddcallback** command, you can specify a script that runs when either of these events occurs. For more information, see the topic *mmaddcallback command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

The file with the policy rules used by **mmapplypolicy** is the one that is currently installed in the file system. It is a good idea for the HSM user to define migration or deletion rules to reduce the usage in

each online storage pool. Migration rules that are defined with a high and low THRESHOLD establish
| the threshold that is used to signal the **lowDiskSpace** event for that pool. Because more than one migration rule can be defined, the threshold for a pool is the minimum of the high thresholds set by the rules for that pool. Each pool has its own threshold. Pools without migration rules do not signal a
| **lowDiskSpace** event.

Backup and restore with storage pools

When you back up data or restore data to a storage pool, consider the following descriptions.

You can use the GPFS ILM tools to backup data for disaster recovery or data archival to an external storage manager such as the TSM Backup-Archive client. When backing up data, the external storage manager must preserve the file name, attributes, extended attributes, and the file data. Among other things, the extended attributes of the file also contain information about the assigned storage pool for the file. When you restore the file, this information is used to assign the storage pool for the file data.

The file data may be restored to the storage pool to which it was assigned when it was backed up or it may be restored to a pool selected by a restore or placement rule using the backed up attributes for the file. GPFS supplies three subroutines that support backup and restore functions with external pools:

- **gpfs_fgetattrs()**
- **gpfs_fputattrs()**
- **gpfs_fputattrswithpathname()**

GPFS exports the extended attributes for a file, including its ACLs, using **gpfs_fgetattrs()**. Included in the extended attributes is the name of the storage pool to which the file has been assigned, as well as file attributes that are used for file placement. When the file is restored the extended attributes are restored using either **gpfs_fputattrs()** or **gpfs_fputattrswithpathname()**.

When a backup application uses **gpfs_fputattrs()** to restore the file, GPFS assigns the restored file to the storage pool with the same name as when the file was backed up. Thus by default, restored files are assigned to the same storage pool they were in when they were backed up. If that pool is not available, GPFS tries to select a pool using the current file placement rules. If that fails, GPFS assigns the file to the system storage pool.

Note: If a backup application uses **gpfs_fputattrs()** to restore a file, it will omit the **RESTORE** RULE.

When a backup application restores the file using **gpfs_fputattrswithpathname()**, GPFS is able to access additional file attributes that may have been used by placement or migration policy rules to select the storage pool for the file. This information includes the UID and GID for the owner, the access time for the file, file modification time, file size, the amount of storage allocated, and the full path to the file. GPFS uses **gpfs_fputattrswithpathname()** to match this information with restore policy rules you define.

In other words, the **RESTORE** rule looks at saved file attributes rather than the current file attributes. The call to **gpfs_fputattrswithpathname()** tries to match the saved information to a **RESTORE** rule. If the **RESTORE** rules cannot match saved attributes, GPFS tries to restore the file to the same storage pool it was in when the file was backed up. If that pool is not available GPFS tries to select a pool by matching placement rules. If that fails, GPFS assigns the file to the system storage pool.

Note: When a **RESTORE** rule is used, and restoring the file to the specified pool would exceed the occupancy percentage defined for that pool, GPFS skips that rule and the policy engine looks for the next rule that matches. While testing for matching rules, GPFS takes into account the specified replication factor and the **KB_ALLOCATED** attribute of the file that is being restored.

The **gpfs_fgetattrs()**, **gpfs_fputattrs()**, and **gpfs_fputattrswithpathname()** subroutines have optional flags that further control the selection of storage pools. For more information, see the topics *gpfs_fgetattrs() subroutine*, *gpfs_fputattrs() subroutine*, and *gpfs_fputattrswithpathname() subroutine* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Working with external lists

External lists, like external pools, generate lists of files. For external pools, the operations on the files correspond to the rule that references the external pool. For external lists, there is no implied operation; it is simply a list of files that match the criteria specified in the policy rule.

External lists must be defined before they can be used. External lists are defined by:

```
RULE EXTERNAL LIST 'ListName' EXEC 'InterfaceScript' [OPTS 'OptionsString'] [ESCAPE 'SpecialCharacters']
```

Where:
- *ListName* defines the name of the external list
- *InterfaceScript* defines the program to be invoked to operate on the list of files
- *OptionsString* is an optional string that, if provided, will be passed to the *InterfaceScript*

See "User-provided program for managing external pools" on page 49.

Example

The following rule defines an external list called `listfiles`:

```
RULE EXTERNAL LIST 'listfiles' EXEC '/var/mmfs/etc/listControl' OPTS '-verbose'
```

In this example:
- `listfiles` is the name of the external list
- `/var/mmfs/etc/listControl` is the location of the executable script that defines the operations on the list of files
- `-verbose` is an optional flag to the `listControl` script

The EXTERNAL LIST rule provides the binding between the lists generated with regular LIST rules and the external program that you want to run with these lists as input. For example, this rule would generate a list of all files that have more than 1 MB of data in an internal storage pool:

```
RULE 'ListLargeFiles' LIST 'listfiles' WHERE KB_ALLOCATED > 1024
```

By default, only user files are included in lists. To include directories, symbolic links, and other file system objects, the DIRECTORIES_PLUS clause must be specified. For example, this rule would generate a list of all objects in the file system.

```
RULE 'ListAllObjects' LIST 'listfiles' DIRECTORIES_PLUS
```

Filesets

In most file systems, a file hierarchy is represented as a series of directories that form a tree-like structure. Each directory contains other directories, files, or other file-system objects such as symbolic links and hard links. Every file system object has a name associated with it, and is represented in the namespace as a node of the tree.

In addition, GPFS utilizes a file system object called a *fileset*. A fileset is a subtree of a file system namespace that in many respects behaves like an independent file system. Filesets provide a means of partitioning the file system to allow administrative operations at a finer granularity than the entire file system:
- Filesets can be used to define quotas on both data blocks and inodes.

- The owning fileset is an attribute of each file and can be specified in a policy to control initial data placement, migration, and replication of the file's data. See "Policies for automating file management" on page 20.
- Fileset snapshots can be created instead of creating a snapshot of an entire file system.

GPFS supports independent and dependent filesets. An independent fileset is a fileset with its own inode space. An inode space is a collection of inode number ranges reserved for an independent fileset. An inode space enables more efficient per-fileset functions, such as fileset snapshots. A dependent fileset shares the inode space of an existing, independent fileset. Files created in a dependent fileset are assigned inodes in the same collection of inode number ranges that were reserved for the independent fileset from which it was created.

When the file system is created, only one fileset, called the *root* fileset, exists. The root fileset is an independent fileset that cannot be deleted. It contains the root directory as well as any system files such as quota files. As new files and directories are created, they automatically become part of the parent directory's fileset. The fileset to which a file belongs is largely transparent for ordinary file access, but the containing fileset can be displayed along with the other attributes of each file using the **mmlsattr -L** command.

The root directory of a GPFS file system is also the root of the root fileset.

Fileset namespace

A newly created fileset consists of an empty directory for the root of the fileset, and it is initially not linked into the file system's namespace. A newly created fileset is not visible to the user until it is attached to the namespace by issuing the **mmlinkfileset** command.

Filesets are attached to the namespace with a special link called a *junction*. A junction is a special directory entry, much like a POSIX hard link, that connects a name in a directory of one fileset (source) to the root directory of another fileset (target). A fileset may be the target of only one junction, so that a fileset has a unique position in the namespace and a unique path to any of its directories. The target of the junction is referred to as the *child fileset*, and a fileset can have any number of children. From the user's viewpoint, a junction always appears as if it were a directory, but the user is not allowed to issue the **unlink** or **rmdir** commands on a junction.

Once a fileset has been created and linked into the namespace, an administrator can unlink the fileset from the namespace by issuing the **mmunlinkfileset** command. This makes all files and directories within the fileset inaccessible. If other filesets were linked below it, the other filesets become inaccessible, but they do remain linked and will become accessible again when the fileset is re-linked. Unlinking a fileset, like unmounting a file system, fails if there are open files. The **mmunlinkfileset** command has a force option to close the files and force the unlink. If there are open files in a fileset and the fileset is unlinked with the force option, future references to those files will result in **ESTALE** errors. Once a fileset is unlinked, it can be re-linked into the namespace at its original location or any other location (it cannot be linked into its children since they are not part of the namespace while the parent fileset is unlinked).

The namespace inside a fileset is restricted to a single, connected subtree. In other words, a fileset has only one root directory and no other entry points such as hard links from directories in other filesets. Filesets are always connected at the root directory and only the junction makes this connection. Consequently, hard links cannot cross fileset boundaries. Symbolic links, of course, can be used to provide shortcuts to any file system object in the namespace.

The root fileset is an exception. The root fileset is attached to the local namespace using the standard **mount** command. It cannot be created, linked, unlinked or deleted using the GPFS fileset commands.

See "Managing filesets" on page 59.

Filesets and quotas

The GPFS quota commands support the **-j** option for fileset block and inode allocation.

The quota limit on blocks and inodes in a fileset are independent of the limits for specific users or groups of users. See these commands:

- **mmdefedquota**
- **mmdefedquotaon**
- **mmdefedquotaoff**
- **mmedquota**
- **mmlsquota**
- **mmquotaoff**
- **mmquotaon**
- **mmrepquota**

In addition, see the description of the **--perfileset-quota** parameter of the following commands:

- **mmchfs**
- **mmcrfs**
- **mmlsfs**

Filesets and storage pools

Filesets are not specifically related to storage pools, although each file in a fileset physically resides in blocks in a storage pool. This relationship is many-to-many; each file in the fileset can be stored in a different user storage pool.

A storage pool can contain files from many filesets. However, all of the data for a particular file is wholly contained within one storage pool.

Using file-placement policies, you can specify that all files created in a particular fileset are to be stored in a specific storage pool. Using file-management policies, you can define how files in a specific fileset are to be moved or deleted during the file's life cycle. See “Policy rules: Terms” on page 23.

Filesets and global snapshots

A GPFS global snapshot preserves the contents of the entire file system, including all its filesets, even unlinked ones.

The state of filesets in the snapshot is unaffected by changes made to filesets in the active file system, such as unlink, link or delete. The saved file system can be accessed through the **.snapshots** directories and the namespace, including all linked filesets, appears as it did when the snapshot was created. Unlinked filesets are inaccessible in the snapshot, as they were in the active file system. However, restoring a snapshot also restores the unlinked filesets, which can then be re-linked and accessed.

If a fileset is included in a global snapshot, it can be deleted but it is not entirely removed from the file system. In this case, the fileset is emptied of all contents and given a status of 'deleted'. The contents of a fileset remain available in the snapshots that include the fileset (that is, through some path containing a **.snapshots** component) even after the fileset is deleted, since all the contents of the fileset are saved when a snapshot is created. The fileset remains in the deleted state until the last snapshot containing it is deleted, at which time the fileset is automatically deleted.

A fileset is included in a global snapshot if the snapshot is created after the fileset was created. Deleted filesets appear in the output of the **mmlsfileset** and **mmlsfileset --deleted** commands, and the **-L** option can be used to display the latest snapshot that includes a fileset.

During a restore from a global snapshot, attributes of filesets included in the snapshot can be altered. The filesets included in the global snapshot are restored to their former state, and newer filesets are deleted. Also, restore may undelete deleted filesets and change linked filesets to unlinked or vice versa. If the name of a fileset was changed since the snapshot was taken, the old fileset name will be restored.

Fileset-level snapshots

Instead of creating a global snapshot of an entire file system, a fileset snapshot can be created to preserve the contents of a single independent fileset plus all dependent filesets that share the same inode space.

If an independent fileset has dependent filesets that share its inode space, then a snapshot of the independent fileset will also include those dependent filesets. In other words, a fileset snapshot is a snapshot of the whole inode space.

Each independent fileset has its own hidden **.snapshots** directory in the root directory of the fileset that contains any fileset snapshots. The **mmsnapdir** command allows setting an option that makes global snapshots also available through **.snapshots** in the root directory of all independent filesets. The **.snapshots** directory in the file system root directory lists both global snapshots and fileset snapshots of the root fileset (the root fileset is an independent fileset). This behavior can be customized with the **mmsnapdir** command.

Fileset snapshot names need not be unique across different filesets, so it is valid to use the same name for fileset snapshots of two different filesets because they will appear under **.snapshots** in two different fileset root directories.

You can restore independent fileset snapshot data and attribute files with the **mmrestorefs** command. For complete usage information, see the topic *mmrestorefs command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Filesets and backup

The **mmbackup** command and TSM are unaware of the existence of filesets. When restoring a file system that had been backed up to TSM, the files are restored to their original path names, regardless of the filesets of which they were originally a part.

TSM has no mechanism to create or link filesets during restore. Therefore, if a file system is migrated to TSM and then filesets are unlinked or deleted, restore or recall of the file system does not restore the filesets.

During a full restore from backup, all fileset information is lost and all files are restored into the root fileset. It is recommended that you save the output of the **mmlsfileset** command to aid in the reconstruction of fileset names and junction locations. Saving **mmlsfileset -L** also allows reconstruction of fileset comments. Both command outputs are needed to fully restore the fileset configuration.

A partial restore can also lead to confusion if filesets have been deleted, unlinked, or their junctions moved, since the backup was made. For example, if the backed up data was in a fileset that has since been unlinked, the restore process puts it into files and directories in the parent fileset. The unlinked fileset cannot be re-linked into the same location until the restored data is moved out of the way. Similarly, if the fileset was deleted, restoring its contents does not recreate the deleted fileset, but the contents are instead restored into the parent fileset.

Since the **mmbackup** command operates by traversing the directory structure, it does not include the contents of unlinked filesets, even though they are part of the file system. If it is desired to include these filesets in the backup, they should be re-linked, perhaps into a temporary location. Conversely, temporarily unlinking a fileset is a convenient mechanism to exclude it from a backup.

Note: It is recommended not to unlink filesets when doing backups. Unlinking a fileset during an **mmbackup** run can cause the following:

- failure to back up changes in files that belong to an unlinked fileset
- expiration of files that were backed up in a previous **mmbackup** run

In summary, fileset information should be saved by periodically recording **mmlsfileset** output somewhere in the file system, where it is preserved as part of the backup process. During restore, care should be exercised when changes in the fileset structure have occurred since the backup was created.

Attention: If you are using the TSM Backup-Archive client you must use caution when you unlink filesets that contain data backed up by TSM. TSM tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to TSM that you deleted the contents of the fileset. Therefore, the TSM Backup-Archive client inactivates the data on the TSM server which may result in the loss of backup data during the expiration process.

Managing filesets

Managing your filesets includes:

- "Creating a fileset"
- "Deleting a fileset" on page 60
- "Linking a fileset" on page 60
- "Unlinking a fileset" on page 60
- "Changing fileset attributes" on page 61
- "Displaying fileset information" on page 61

Creating a fileset

Filesets are created with the **mmcrfileset** command.

By default, filesets are created as dependent filesets that share the root's inode space. The **--inode-space** *ExistingFileset* option can be used to create a dependent fileset that will share inode space with an existing fileset. The *ExistingFileset* specified can be **root** or any other independent fileset, but cannot be the reserved keyword **new**. The **--inode-space new** option can be used to create an independent fileset with its own dedicated inode space.

A newly created fileset consists of an empty directory for the root of the fileset and it is initially not linked into the existing namespace. Consequently, a new fileset is not visible, nor can files be added to it, but the fileset name is valid and the administrator can establish quotas on it, or policies for it. The administrator must link the fileset into its desired location in the file system's namespace by issuing the **mmlinkfileset** command in order to make use of it.

After the fileset is linked, the administrator can change the ownership and permissions for the new fileset's root directory, which default to **root** and 0700, to allow users access to it. Files and directories copied into, or created within, the fileset's directory will become part of the new fileset.

Fileset names must follow these conventions:

- Are character strings and must be less than 256 characters in length.
- Must be unique within a file system.
- The name **root** is reserved for the fileset of the files system's root directory.

For complete usage information, see the topics *mmcrfileset command* and *mmlinkfileset command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Deleting a fileset

Filesets are deleted with the **mmdelfileset** command.

There are several notes to keep in mind when deleting filesets:

- The root fileset cannot be deleted.
- A fileset that is not empty cannot be deleted unless the **-f** flag is specified.
- A fileset that is currently linked into the namespace cannot be deleted until it is unlinked with the **mmunlinkfileset** command.
- A dependent fileset can be deleted at any time.
- An independent fileset cannot be deleted if it has any dependent filesets or fileset snapshots.
- Deleting a dependent fileset that is included in a fileset or global snapshot removes it from the active file system, but it remains part of the file system in a deleted state.
- Deleting an independent fileset that is included in any global snapshots removes it from the active file system, but it remains part of the file system in a deleted state.
- A fileset in the deleted state is displayed in the **mmlsfileset** output with the fileset name in parenthesis. If the **-L** flag is specified, the latest including snapshot is also displayed. The **--deleted** option of the **mmlsfileset** command can be used to display only deleted filesets.
- The contents of a deleted fileset are still available in the snapshot, through some path name containing a **.snapshots** component, because it was saved when the snapshot was created.
- When the last snapshot that includes the fileset has been deleted, the fileset is fully removed from the file system.

For complete usage information, see the topics *mmdelfileset command*, *mmlsfileset command*, and *mmunlinkfileset command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Linking a fileset

After the fileset is created, a junction must be created to link it to the desired location in the file system's namespace using the **mmlinkfileset** command.

The file system must be mounted in order to link a fileset. An independent fileset can be linked into only one location anywhere in the namespace, specified by the *JunctionPath* parameter:

- The root directory
- Any subdirectory
- The root fileset or to any other fileset

A dependent fileset can only be linked inside its own inode space.

If *JunctionPath* is not specified, the junction is created in the current directory and has the same name as the fileset being linked. After the command completes, the new junction appears as an ordinary directory, except that the user is not allowed to unlink or delete it with the **rmdir** command it. The user can use the **mv** command on the directory to move to a new location in the parent fileset, but the **mv** command is not allowed to move the junction to a different fileset.

For complete usage information, see the topic *mmlinkfileset command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Unlinking a fileset

A junction to a fileset is removed with the **mmunlinkfileset** command, which unlinks the fileset only from the active directory namespace. The linked or unlinked state of a fileset in a snapshot is unaffected. The unlink fails if there are files open in the fileset, unless the **-f** option is specified. The root fileset cannot be unlinked.

After issuing the **mmunlinkfileset** command, the fileset can be re-linked to a different parent using the **mmlinkfileset** command. Until the fileset is re-linked, it is not accessible.

Note: If run against a file system that has an unlinked fileset, **mmapplypolicy** will not traverse the unlinked fileset.

Attention: If you are using the TSM Backup-Archive client you must use caution when you unlink filesets that contain data backed up by TSM. TSM tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to TSM that you deleted the contents of the fileset. Therefore, the TSM Backup-Archive client inactivates the data on the TSM server which may result in the loss of backup data during the expiration process.

For complete usage information, see the topic *mmunlinkfileset command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Changing fileset attributes

To change a fileset's junction, you have to first unlink the fileset using the **mmunlinkfileset** command, and then create the new junction using the **mmlinkfileset** command.

To change the attributes of an existing fileset, including the fileset name, use the **mmchfileset** command.

Note: In an HSM-managed file system, moving or renaming migrated files between filesets will result in recalling of the date from the TSM server.

For complete usage information, see the topics *mmchfileset command*, *mmlinkfilesetcommand*, and *mmunlinkfileset command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Displaying fileset information

Fileset status and attributes are displayed with the **mmlsfileset** command.

Some of the attributes displayed include:

- Name of the fileset.
- Fileset identifier of the fileset.
- Junction path to the fileset.
- Status of the fileset.
- Root inode number of the fileset.
- Path to the fileset (if linked).
- Inode space.
- User provided comments (if any).

For complete usage information, see the topic *mmlsfileset command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

To display the name of the fileset that includes a given file, run the **mmlsattr** command and specify the **-L** option. For complete usage information, see the topic *mmlsattr command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Immutability and appendOnly features

To prevent files from being changed or deleted unexpectedly, GPFS provides immutability and appendOnly restrictions.

Applying immutability and appendOnly restrictions to individual files or to directories

You can apply immutability and appendOnly restrictions either to individual files within a fileset or to a directory.

An immutable file cannot be changed or renamed. An appendOnly file allows append operations, but not delete, modify, or rename operations.

An immutable directory cannot be deleted or renamed, and files cannot be added or deleted under such a directory. An appendOnly directory allows new files or subdirectories to be created with 0 byte length; all such new created files and subdirectories are marked as appendOnly automatically.

The **immutable** flag and the **appendOnly** flag can be set independently. If both immutability and appendOnly are set on a file, immutability restrictions will be in effect.

To set or unset these attributes, use the following command options:

mmchattr -i yes|no
Sets or unsets a file to or from an immutable state.

-i yes Sets the **immutable** attribute of the file to **yes**.

-i no Sets the **immutable** attribute of the file to **no**.

mmchattr -a yes|no
Sets or unsets a file to or from an appendOnly state.

-a yes Sets the **appendOnly** attribute of the file to **yes**.

-a no Sets the **appendOnly** attribute of the file to **no**.

Note: Before an immutable or appendOnly file can be deleted, you must change it to mutable or set appendOnly to **no** (by using the **mmchattr** command).

Storage pool assignment of an immutable or appendOnly file can be changed; an immutable or appendOnly file is allowed to transfer from one storage pool to another.

To display whether or not a file is immutable or appendOnly, issue this command:

```
mmlsattr -L myfile
```

The system displays information similar to the following:

```
file name:            myfile
metadata replication: 2 max 2
data replication:     1 max 2
immutable:            no
appendOnly:           no
flags:
storage pool name:    sp1
fileset name:         root
snapshot name:
creation Time:        Wed Feb 22 15:16:29 2012
Misc attributes:      ARCHIVE
```

The effects of file operations on immutable and appendOnly files

Once a file has been set as immutable or appendOnly, the following file operations and attributes work differently from the way they work on regular files:

delete An immutable or appendOnly file cannot be deleted.

modify/append
An immutable file cannot be modified or appended

An appendOnly file cannot be modified, but it can be appended..

Note: The immutable and appendOnly flag check takes effect after the file is closed; therefore, the file can be modified if it is opened before the file is changed to immutable.

mode An immutable or appendOnly file's mode cannot be changed.

ownership, acl
These attributes cannot be changed for an immutable or appendOnly file.

extended attributes
These attributes cannot be added, deleted, or modified for an immutable or appendOnly file.

timestamp
The timestamp of an immutable or appendOnly file can be changed.

directory
If a directory is marked as immutable, no files can be created, renamed, or deleted under that directory. However, a subdirectory under an immutable directory remains mutable unless it is explicitly changed by **mmchattr**.

If a directory is marked as appendOnly, no files can be renamed or deleted under that directory. However, 0 byte length files can be created.

The following table shows the effects of file operations on an immutable file or an appendOnly file:

Table 4. The effects of file operations on an immutable file or an appendOnly file

Operation	immutable	appendOnly
Add, **delete**, **modify**, or **rename**	No	No
Append	No	Yes
Change ownership, **mode**, or **acl**	No	No
Change atime, **mtime**, or **ctime**	Yes	Yes
Add, **delete**, or **modify extended attributes**	Disallowed by external methods such as **setfattr**. Allowed internally for **dmapi**, **directio**, and others.	Same as for immutable.
Create a file under an immutable or appendOnly directory	No	Yes, 0 byte length only
Rename or **delete** a file under an immutable or appendOnly directory	No	No
Modify a mutable file under an immutable directory	Yes	Not applicable
Set an immutable file back to mutable	Yes	Not applicable
Set an appendOnly file back to a non-appendOnly state	Not applicable	Yes

Fileset-level integrated archive manager (IAM) modes

Fileset-level integrated archive manager (IAM) modes can be used to modify some of the file-operation restrictions that normally apply to immutable files (see Table 5 on page 64. The **mmchfileset --iam-mode** parameter is used to set the IAM mode; for details, see the **mmchfileset** command description in *IBM Spectrum Scale: Administration and Programming Reference* .

For more information, see the topic *mmchfilesetcommand* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Table 5. IAM modes and their effect on immutable file operations

File Operation	Regular Mode	Advisory Mode	Noncompliant Mode	Compliant Mode
Modify	No	No	No	No
Append	No	No	No	No
Rename	No	No	No	No
Change ownership, acl	No	No	No	No
Change mode	No	No	No	No
Change **atime**, **mtime**, **ctime**	Yes	**mtime** and **ctime** can be changed. **atime** is overloaded by expiration time. Expiration time is changed by using the **mmchattr --expiration-time** parameter (or its alternative form, **mmchattr -E**), or touch will be shown by using stat as **atime**.	Same as advisory mode	Same as advisory mode
Add, delete, or modify extended attributes.	Not allowed for external methods such as **setfattr**. Allowed internally for dmapi, directio, and so on.	Yes	Yes	Yes
Create, rename, or delete under an immutable directory	No	No	No	No
Modify mutable files under an immutable directory.	Yes	Yes	Yes	Yes
Retention rule enforced	No retention rule, cannot delete immutable files	No	Yes	Yes
Set ExpirationTime backwards	Yes	Yes	Yes	No
Delete an immutable file	No	Yes, always	Yes, only when expired	Yes, only when expired
Set an immutable file back to mutable	Yes	No	No	No
Allow hardlink	Yes	No	No	No
Rename or delete a non-empty directory	Yes	No	No	No

Table 5. IAM modes and their effect on immutable file operations (continued)

File Operation	**Regular Mode**	**Advisory Mode**	**Noncompliant Mode**	**Compliant Mode**
Remove user write permission to change a file to immutable	No	Yes	Yes	Yes
Display expiration time instead of **atime** for stat call	No	Yes	Yes	Yes
Set a directory to be immutable	Yes	No	No	No

Chapter 3. Creating and maintaining snapshots of file systems

A snapshot of an entire GPFS file system can be created to preserve the contents of the file system at a single point in time. Snapshots of the entire file system are also known as global snapshots. The storage overhead for maintaining a snapshot is keeping a copy of data blocks that would otherwise be changed or deleted after the time of the snapshot.

Snapshots of a file system are read-only; changes can only be made to the active (that is, normal, non-snapshot) files and directories.

The snapshot function allows a backup or mirror program to run concurrently with user updates and still obtain a consistent copy of the file system as of the time that the snapshot was created. Snapshots also provide an online backup capability that allows easy recovery from common problems such as accidental deletion of a file, and comparison with older versions of a file.

Notes:
1. Because snapshots are not copies of the entire file system, they should not be used as protection against media failures. For information about protection against media failures, see the topic *Recoverability considerations* in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.
2. Fileset snapshots provide a method to create a snapshot of an independent fileset instead of the entire file system. For more information about fileset snapshots, see "Fileset-level snapshots" on page 58.
3. A snapshot of a file creates a new file that captures the user data and user attributes from the original. The snapshot file is independent from the original file. For DMAPI managed file systems, the snapshot of a file is not automatically managed by DMAPI, regardless of the state of the original file. The DMAPI attributes from the original file are not inherited by the snapshot. For more information about DMAPI restrictions for GPFS, see the *IBM Spectrum Scale: Data Management API Guide*.
4. When snapshots are present, deleting files from the active file system does not always result in any space actually being freed up; rather, blocks may be pushed to the previous snapshot. In this situation, the way to free up space is to delete the oldest snapshot. Before creating new snapshots, it is good practice to ensure that the file system is not close to being full.
5. The use of clones functionally provides writable snapshots. See Chapter 4, "Creating and managing file clones," on page 75.

Management of snapshots of a GPFS file system includes:
- "Creating a snapshot"
- "Listing snapshots" on page 68
- "Restoring a file system from a snapshot" on page 69
- "Reading a snapshot with the policy engine" on page 70
- "Linking to a snapshot" on page 70
- "Deleting a snapshot" on page 71

Creating a snapshot

Use the **mmcrsnapshot** command to create a snapshot of an entire GPFS file system at a single point in time. Snapshots appear in the file system tree as hidden subdirectories of the root.

Global snapshots appear in a subdirectory in the root directory of the file system, whose default name is **.snapshots**. If you prefer to access snapshots from each directory rather than traversing through the root directory, you can use an invisible directory to make the connection by issuing the **mmsnapdir** command. See "Linking to a snapshot" on page 70 for more information.

A snapshot of the file system, *Device*, is identified by a *SnapshotName* name on the **mmcrsnapshot** command. For example, given the file system **fs1** to create a snapshot **snap1**, enter:

```
mmcrsnapshot fs1 snap1
```

The output is similar to this:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap1 created with id 1.
```

Before issuing the command, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If a second snapshot were to be created at a later time, the first snapshot would remain as is. Snapshots are only made of active file systems, not existing snapshots. For example:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap2 created with id 2.
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3

/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userA/file2
/fs1/.snapshots/snap2/userA/file3
```

For complete usage information, see the topic *mmcrsnapshot command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Listing snapshots

Use the **mmlssnapshot** command to display existing snapshots of a file system and their attributes.

The **-d** option displays the amount of storage used by a snapshot. GPFS quota management does not take the data blocks used to store snapshots into account when reporting on and determining if quota limits have been exceeded. This is a slow operation and its usage is suggested for problem determination only.

For example, to display the snapshot information for the file system **fs1** with additional storage information, issue this command:

```
mmlssnapshot fs1 -d
```

The system displays information similar to:

```
Snapshots in file system fs1: [data and metadata in KB]
Directory                SnapId    Status          Created                         Data  Metadata
snap1                    1         Valid           Fri Oct 17 10:56:22 2003           0       512
```

For complete usage information, see the topic *mmlssnapshot command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Restoring a file system from a snapshot

Use the **mmrestorefs** command to restore user data and attribute files in an active file system from a snapshot.

Prior to issuing the **mmrestorefs** command, ensure that the file system is mounted. When restoring from an independent fileset snapshot, ensure that the fileset is in linked state.

Existing snapshots, including the one being used in the restore, are not modified by the **mmrestorefs** command. To obtain a snapshot of the restored file system, you must issue the **mmcrsnapshot** command to capture it before issuing the **mmrestorefs** command again.

As an example, suppose that you have a directory structure similar to the following:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If the directory **userA** is then deleted, the structure becomes similar to this:

```
/fs1/file1
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

The directory **userB** is then created using the inode originally assigned to **userA**, and another snapshot is taken:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
Writing dirty data to disk.
Quiescing all file system operations.
Writing dirty data to disk again.
Snapshot snap2 created with id 2.
```

The resulting directory structure is similar to the following:

```
/fs1/file1
/fs1/userB/file2b
/fs1/userB/file3b
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
```

```
/fs1/.snapshots/snap1/userA/file3
/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

The file system is then restored from **snap1**:

```
mmrestorefs fs1 snap1
```

The resulting directory structure is similar to the following:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

For complete usage information, see the topic *mmrestorefs command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Reading a snapshot with the policy engine

You can use the policy engine to read the contents of a snapshot for backup purposes. The **mmapplypolicy** command provides the **-S** option to specify the snapshot during a policy run. Instead of matching rules to the active file system, the policy engine matches the rules against files in the snapshot.

Notes:

1. Snapshots are read-only. Policy rules such as MIGRATE or DELETE that make changes or delete files cannot be used with a snapshot.
2. An instance of **mmapplypolicy** can only scan one snapshot. Directing it at the `.snapshots` directory itself will result in a failure.

For complete usage information, see the topic *mmapplypolicy command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Linking to a snapshot

Snapshot root directories appear in a special **.snapshots** directory under the file system root.

If you prefer to link directly to the snapshot rather than always traverse the root directory, you can use the **mmsnapdir** command with the **-a** option to add a **.snapshots** subdirectory to all directories in the file system. These **.snapshots** subdirectories will contain a link into the corresponding directory for each snapshot that includes the directory in the active file system.

Unlike **.snapshots** in the root directory, however, the **.snapshots** directories added by the **-a** option of the **mmsnapdir** command are invisible in the sense that the **ls** command or **readdir()** function does not return **.snapshots**. This is to prevent recursive file system utilities such as **find** or **tar** from entering into the snapshot tree for each directory they process. For example, if you enter **ls -a /fs1/userA**, the **.snapshots** directory is not listed. However, you can enter **ls /fs1/userA/.snapshots** or **cd /fs1/userA/.snapshots** to confirm that **.snapshots** is present. If a user wants to make one of their snapshot directories more visible, it is suggested to create a symbolic link to **.snapshots**.

The inode numbers that are used for and within these special **.snapshots** directories are constructed dynamically and do not follow the standard rules. These inode numbers are visible to applications through standard commands, such as **stat**, **readdir**, or **ls**. The inode numbers reported for these directories can also be reported differently on different operating systems. Applications should not expect consistent numbering for such inodes.

Specifying the **-r** option on the **mmsnapdir** command reverses the effect of the **-a** option, and reverts to the default behavior of a single **.snapshots** directory in the root directory.

The **-s** option allows you to change the name of the **.snapshots** directory. For complete usage information, see the topic *mmsnapdir command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

To illustrate this point, assume that a GPFS file system called **fs1**, which is mounted at **/fs1**, has one snapshot called **snap1**. The file system might appear similar to this:

```
/fs1/userA/file2b
/fs1/userA/file3b
/fs1/.snapshots/snap1/userA/file2b
/fs1/.snapshots/snap1/userA/file3b
```

To create links to the snapshots from each directory, and instead of **.snapshots**, use the name **.links**, enter:

```
mmsnapdir fs1 -a -s .links
```

After the command completes, the directory structure would appear similar to:

```
/fs1/userA/file2b
/fs1/userA/file3b
/fs1/userA/.links/snap1/file2b
/fs1/userA/.links/snap1/file3b

/fs1/.links/snap1/userA/file2b
/fs1/.links/snap1/userA/file3b
```

To delete the links, issue:

```
mmsnapdir fs1 -r
```

After the command completes, the directory structure is similar to the following:

```
/fs1/userA/file2b
/fs1/userA/file3b

/fs1/.links/snap1/userA/file2b
/fs1/.links/snap1/userA/file3b
```

For complete usage information, see the topic *mmsnapdir command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Deleting a snapshot

Use the **mmdelsnapshot** command to delete GPFS snapshots of a file system.

For example, to delete **snap1** for the file system **fs1**, enter:

```
mmdelsnapshot fs1 snap1
```

The output is similar to this:

```
Invalidating snapshot files...
Deleting snapshot files...
 100.00 % complete on Tue Feb 28 10:40:59 2012
Delete snapshot snap1 complete, err = 0
```

For complete usage information, see the topic *mmdelsnapshot command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Managing snapshots with IBM Spectrum Scale GUI

Use **Files** > **Snapshots** page in the IBM Spectrum Scale GUI to manage snapshots through GUI.

Snapshots can be used in environments where multiple recovery points are necessary. A snapshot can be taken of file system or fileset data and then the data can be recovered from the snapshot if the production data becomes unavailable.

Note:
- Snapshots are read-only; changes can be made only to the normal and active files and directories, not to the snapshot.
- When a snapshot of an independent fileset is taken, only nested dependent filesets are included in the snapshot.

Scheduling snapshot creation by using snapshot rules

You can either manually create the snapshots or create snapshot rules to automate the snapshot creation and retention.

To manually create a snapshot, click **Create Snapshot** in the Snapshots page and enter the required details under the **Manual** tab of the Create Snapshot window. Click **Create** after entering the details.

By creating a snapshot rule, you can automate the snapshot creation and retention. That is, in a snapshot rule you can specify a frequency in which the snapshots must be created and the number of snapshots that must be retained for a period. A retention policy can have the following parameters:
- Hours
- Days
- Weeks
- Months

User can specify values for each. The following table provides an example for the values that are specified against these parameters.

Table 6. Example for retention period

Hours	Days	Weeks	Months
5	3	3	4

This means, only five snapshots are stored in every hour, three in a day, three in a week, and four in a month. Based on the retention period, the snapshots are cleared from the storage in first in first out (FIFO) manner. Snapshots are cleared from the storage only once per day due to performance issues. The retention policy helps to avoid unwanted storage of snapshots that result in waste of storage resources.

To schedule snapshot creation and retention, perform the following steps:
1. Go to **Files > Snapshots**.
2. Click **Create Snapshot**.
3. In the Create Snapshot window, enter the path of the file system or independent fileset for which you need to create snapshots.
4. In the **Snapshot name** field, specify the name of the snapshot.
5. Click **Create Rule** to schedule the snapshot creation and retention. Create Snapshot Rule window is displayed.
6. In the **Name** field, type the name of the snapshot scheduling rule.

7. In the **Frequency** field, select the frequency in which you need to create snapshot. You need to enter some more details based on the value that is selected in the **Frequency** field. For example, if value selected is **Multiple Times an Hour**, select the minutes of the hour in which you need to create snapshots.
8. In the **Retention** fields, specify the number of snapshots that must be retained in a time period.
9. In the **Prefix** field, specify a prefix to be added with the name of the snapshots that are created with this rule.
10. Click **OK** to save the changes.
11. In the **Create Snapshot** window, click **Create** to create the snapshot creation schedule and retention policy for the snapshots.

If you do not specify a name for the snapshot, the default name is given. The default snapshot ID is generated at the creation time by using the format "*@GMT-yyyy.MM.dd-HH.mm.ss*". If this option is given and the "*@GMT-date-time*" format is omitted, then this snapshot will not be identifiable by Windows VSS and the file restore is not possible by that method. Avoid white spaces, double and single quotation marks, the parentheses (), the star *, forward slash /, and backward slash \.

Deleting snapshots

Use caution when removing snapshots that are associated with backups or when removing any file system snapshots that are used by any other component, such as Network Data Management Protocol (NDMP) or asynchronous replication.

For example, when you remove an NDMP snapshot, an NDMP backup that is defined to use the removed NDMP snapshot fails. If you remove an NDMP snapshot and the tracking file still exists, the NDMP backup attempts to use the removed snapshot, unless the snapshot is expired. To clean up all of the temporary files after you delete an NDMP snapshot, you must deactivate and reactivate NDMP for the associated NDMP node group.

The deletion of file set snapshots is an I/O intensive operation and the initiation of many of these concurrently might cause contention for I/O resources, resulting in a slowdown of the entire system. In extreme cases, this problem might look like a hung system and some operations might fail. The exact thresholds for this slowdown depend on a number of variables including write and read ratios, I/O capacity, and other concurrent operations. Due to these performance issues, snapshots are deleted from the storage only once per day. The snapshot deletion time is set to 02:30 AM everyday. You can change this value by modifying the **`SNAPHOT_DELETE_TIME`** parameter that is located in the `/usr/lpp/mmfs/gui/conf/gpfsgui.properties` file. You need to restart the GUI service for the changes to take effect.

Chapter 4. Creating and managing file clones

A file clone is a writable snapshot of an individual file. File clones can be used to provision virtual machines by creating a virtual disk for each machine by cloning a common base image. A related usage is to clone the virtual disk image of an individual machine as part of taking a snapshot of the machine state.

Cloning a file is similar to creating a copy of a file, but the creation process is faster and more space efficient because no additional disk space is consumed until the clone or the original file is modified. Multiple clones of the same file can be created with no additional space overhead. You can also create clones of clones.

Management of file clones in a GPFS file system includes:
- "Creating file clones"
- "Listing file clones" on page 76
- "Deleting file clones" on page 77
- "File clones and disk space management" on page 77
- "File clones and snapshots" on page 77

Creating file clones

File clones can be created from a regular file or a file in a snapshot using the **mmclone** command.

Creating a file clone from a regular file is a two-step process using the **mmclone** command with the **snap** and **copy** keywords:

1. Issue the **mmclone snap** command to create a read-only snapshot of the file to be cloned. This read-only snapshot becomes known as the clone parent. For example, the following command creates a clone parent called **snap1** from the original file **file1**:

   ```
   mmclone snap file1 snap1
   ```

   Alternately, if only one file is specified with the **mmclone snap** command, it will convert the file to a read-only clone parent without creating a separate clone parent file. When using this method to create a clone parent, the specified file cannot be open for writing or have hard links. For example, the following command converts **file1** into a clone parent.

   ```
   mmclone snap file1
   ```
2. Issue the **mmclone copy** command to create a writable clone from a clone parent. For example, the following command creates a writable file clone called **file2** from the clone parent **snap1**:

   ```
   mmclone copy snap1 file2
   ```

Creating a file clone where the source is in a snapshot only requires one step using the **mmclone** command with the **copy** keyword. For example, the following command creates a writable file clone called **file3.clone** from a file called **file3** in a snapshot called **snap2**:

```
mmclone copy /fs1/.snapshots/snap2/file3 file3.clone
```

In this case, **file3** becomes the clone parent.

Note: Extended attributes of clone parents are not passed along to file clones.

After a clone has been created, the clone and the file that it was cloned from are interchangeable, which is similar to a regular copy (**cp**) command. The file clone will have a new inode number and attributes that can be modified independently of the original file.

Additional clones can be created from the same clone parent by issuing additional **mmclone copy** commands, for example:

```
mmclone copy snap1 file3
```

File clones of clones can also be created, as shown in the following example:

```
mmclone snap file1 snap1
mmclone copy snap1 file2
echo hello >> file2
mmclone snap file2 snap2
mmclone copy snap2 file3
```

The echo command updates the last block of file clone **file2**. When **file2** is snapped to **snap2**, the **mmclone snap** operation is performed as described previously. When a block in **file3** is read, the clone parent inode is found first. For the case of the last block, with the **hello** text, the disk address will be found in **snap2**. However, for other blocks, the disk address will be found in **snap1**.

See the **mmclone** command for complete usage information.

Listing file clones

Use the **mmclone** command to display status for specified files.

The **show** keyword of the **mmclone** command provides a report to determine the current status of one or more files. When a file is a clone, the report will show the parent inode number. When a file was cloned from a file in a snapshot, **mmclone show** displays the snapshot and fileset information.

Consider the following scenario:

1. The **ls** command is issued to show all **img** files in the current directory:
   ```
   ls -ils *.img
   ```
   The system displays output similar to the following:
   ```
   148485 5752576 -rw-r--r-- 1 root root 21474836480 Jan  9 16:19 test01.img
   ```
2. A file clone is then created with the following commands:
   ```
   mmclone snap test01.img base.img
   mmclone copy base.img test02.img
   ```
3. After the file clone is created, the **mmclone show** command is issued to show information about all **img** files in the current directory:
   ```
   mmclone show *.img
   ```
   The system displays output similar to the following:
   ```
   Parent  Depth   Parent inode   File name
   ------  -----  --------------  ---------
      yes      0                  base.img
       no      1          148488  test01.img
       no      1          148488  test02.img
   ```
4. A subsequent **ls** command would display output similar to the following:
   ```
   # ls -ils *.img
   148488 5752576 -rw-r--r-- 3 root root 21474836480 Jan  9 16:25 base.img
   148485       0 -rw-r--r-- 1 root root 21474836480 Jan  9 16:19 test01.img
   148480       0 -rw-r--r-- 1 root root 21474836480 Jan  9 16:25 test02.img
   ```

See the **mmclone** command for complete usage information.

Deleting file clones

There is no explicit GPFS command available for deleting file clones. File clones can be deleted using a regular delete (**rm**) command. Clone parent files cannot be deleted until all file clone copies of the parent have been deleted and all open file handles to them have been closed.

Note: There is a brief period of time, immediately following the deletion of the file clone copies, when deletion of the parent can fail because the clone copy deletions are still running in the background.

Splitting file clones from clone parents

Use the **mmclone** command to split a file clone from a clone parent.

File clones can be split from their clone parents in one of two ways:

- Using the **mmclone redirect** command to split the file clone from the immediate clone parent only. The clone child remains a file clone, but the clone parent can be deleted.
- Using the **mmclone split** command to split the file clone from all clone parents. This converts the former clone child to a regular file. The clone parent does not change.

See the **mmclone** command for complete usage information.

File clones and disk space management

File clones have several considerations that are related to disk space management.

- Replication and storage pools

  Each file clone has its own inode, so attributes and permissions for each clone can be set independently. For example, timestamps (atime, mtime, and ctime) are maintained separately for each clone. Clone parent attributes must be changed separately. If different clones have different values for replication or storage pool, it is not possible for every one of the clones to have all data blocks readable through that clone to be replicated and placed consistent with its replication and pool settings. Thus, changes to replication and storage pool only apply to blocks added to the clone and leave the clone parent unchanged.
- Clone ownership, block counts, and quotas

  Creating a clone parent requires read access to the file being cloned. The person creating the clone parent does not have to be the owner of the original file, but will become the owner of the new clone parent. The block count and disk quota of the original file will be transferred to the new clone parent inode and then set to zero in the original file. Any blocks allocated by copy-on-write of a clone file will be added to the block count in the clone inode, with quota charged against the owner of the file. If even a single byte of data in a clone child is changed, the entire block will be copied from the parent.
- Clones and DMAPI

  Clone parent files and clone copy files will be preserved across migrations and recalls to and from tape.

File clones and snapshots

When a snapshot is created and a file clone is subsequently updated, the previous state of the file clone will be saved in the snapshot.

When reading a file clone in the snapshot, the system will distinguish between the states of the clone:

- The data block has not been modified since the snapshot was taken, so look for the data in the same file in the next most recent snapshot or active file system.
- The file clone was updated, which indicates that the corresponding data block should be found in the clone parent. The system will look for the data in the clone parent within the same snapshot.

When a snapshot has file clones, those file clones should be deleted or split from their clone parents prior to deleting the snapshot. See "Deleting file clones" on page 77 and "Splitting file clones from clone parents" on page 77 for more information. A policy file can be created to help determine if a snapshot has file clones. See "File clones and policy files" for more information.

File clones and policy files

Policy files can be created to examine clone attributes.

The following clone attributes can be examined in a policy file:
- The depth of the clone tree.
- If file is an immutable clone parent.
- The fileset ID of the clone parent.
- The inode number of the clone parent for the file.
- If the clone parent is in a snapshot.
- The snapshot ID of the clone parent.

See "File attributes in SQL expressions" on page 28 for more information about the clone attributes available for policy files.

The following example shows a policy file that can be created for displaying clone attributes for all files:

```
RULE EXTERNAL LIST 'x' EXEC ''
RULE 'nonClone' LIST 'x' SHOW('nonclone') WHERE Clone_Parent_Inode IS NULL
RULE 'normalClone' LIST 'x' SHOW(
  'inum ' || varchar(Clone_Parent_Inode) ||
  ' par ' || varchar(Clone_Is_Parent) ||
  ' psn ' || varchar(Clone_Parent_Is_Snap) ||
  ' dep ' || varchar(Clone_Depth))
  WHERE Clone_Parent_Inode IS NOT NULL AND Clone_Parent_Is_Snap == 0
RULE 'snapClone' LIST 'x' SHOW(
  'inum ' || varchar(Clone_Parent_Inode) ||
  ' par ' || varchar(Clone_Is_Parent) ||
  ' psn ' || varchar(Clone_Parent_Is_Snap) ||
  ' dep ' || varchar(Clone_Depth) ||
  ' Fid ' || varchar(Clone_Parent_Fileset_Id) ||
  ' snap ' || varchar(Clone_Parent_Snap_Id))
  WHERE Clone_Parent_Inode IS NOT NULL AND Clone_Parent_Is_Snap != 0
```

If this policy file was called **pol.file**, the following command would display the clone attributes:

```
mmapplypolicy fs0 -P pol.file -I defer -f pol -L 0
```

Chapter 5. Scale Out Backup and Restore (SOBAR)

Scale Out Backup and Restore (SOBAR) is a specialized mechanism for data protection against disaster only for GPFS file systems that are managed by Tivoli Storage Manager (TSM) Hierarchical Storage Management (HSM).

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

To protect a file system against disaster the following steps must be taken to ensure all data is safely stored in a second location:

1. Record the file system configuration with the **mmbackupconfig** command.
2. Ensure all file data is *pre-migrated* (see "Pre-migrating files with external storage pools" on page 52 for more information).
3. Perform a metadata image backup with the **mmimgbackup** command.

The **mmbackupconfig** command must be run prior to running the **mmimgbackup** command. No changes to file system configuration, filesets, quotas, or other settings should be done between running the **mmbackupconfig** command and the **mmimgbackup** command. To recover from a disaster, the **mmrestoreconfig** command must be run prior to running the **mmimgrestore** command. After restoring the image data and adjusting quota settings, the file system can be mounted read-write, and the HSM system re-enabled to permit file data recall. Users may be permitted to access the file system, and/or the system administrator can manually recall file data with the Tivoli HSM command **dsmrecall**.

These commands cannot be run from a Windows node.

Backup procedure with SOBAR

This section provides a detailed example of the backup procedure used with SOBAR.

Throughout these procedures, the sample file system used is called **smallfs**. Where appropriate, replace this value with your file system name.

1. Backup the cluster configuration information.

   The cluster configuration must be backed up by the administrator. The minimum cluster configuration information needed is: IP addresses, node names, roles, quorum and server roles, cluster-wide configuration settings from **mmchconfig**, cluster manager node roles, remote shell configuration, mutual **ssh** and **rsh** authentication setup, and the cluster UID. More complete configuration information can be found in the **mmsdrfs** file.
2. Preserve disk configuration information.

   Disk configuration must also be preserved in order to recover a file system. The basic disk configuration information needed, for a backup intended for disaster recovery, is the number of disk volumes that were previously available and the sizes of those volumes. In order to recover from a complete file system loss, at least as much disk space as was previously available will be needed for restoration. It is only feasible to restore the image of a file system onto replacement disks if the disk volumes available are of similar enough sizes to the originals that all data can be restored to the new disks. At a minimum, the following disk configuration information is needed:
   - Disk device names
   - Disk device sizes
   - The number of disk volumes
   - NSD server configuration
   - Disk RAID configurations

- Failure group designations
- The **mmsdrfs** file contents

3. Backup the GPFS file system configuration information.

   In addition to the disks, the file system built on those volumes has configuration information that can be captured using the **mmbackupconfig** command. This information includes block size, replication factors, number and size of disks, storage pool layout, filesets and junction points, policy rules, quota information, and a number of other file system attributes. The file system configuration information can be backed up into a single file using a command similar to the following:

   ```
   mmbackupconfig smallfs -o /tmp/smallfs.bkpcfg.out925
   ```

4. Pre-migrate all newer file data into secondary storage.

   File contents in a space-managed GPFS will reside in secondary storage managed by the HSM. In the case of Tivoli Storage Manager HSM, disk and tape pools will typically hold the offline images of migrated files. HSM can also be used to pre-migrate all newer file data into secondary storage, so that all files will have either a migrated or pre-migrated status (**XATTR**) recorded, and their current contents are copied or updated into the secondary storage. The TSM command **dsmmigrate** can be used as follows:

   ```
   dsmmigrate -Premigrate -Recursive /smallfs
   ```

   To optionally check the status of the files that were pre-migrated with the previous command, use the following command:

   ```
   dsmls /smallfs/*
   ```

5. Create a global snapshot of the live file system, to provide a quiescent image for image backup, using a command similar to the following:

   ```
   mmcrsnapshot smallfs smallfssnap
   ```

6. Choose a staging area in which to save the GPFS metadata image files.

   The image backup process stores each piece of the partial file system image backup in its own file in the shared work directory typically used by policy runs. These files can become quite large depending on the number of files in the file system. Also, because the file system holding this shared directory must be accessible to every node participating in the parallel backup task, it might also be a GPFS file system. It is imperative that the staging directory chosen be accessible to both the **tsapolicy** archiver process and the TSM Backup-Archive client. This staging directory is specified with the **-g** option of the **mmimgbackup** command.

7. Backup the file system image.

   The following command will back up an image of the GPFS metadata from the file system using a parallel policy run with the default TSM backup client to backup the file system metadata image:

   ```
   mmimgbackup smallfs -S smallfssnap -g /u/user/backup -N aixnodes
   ```

   The metadata of the file system, the directories, inodes, attributes, symlinks, and so on are all captured in parallel by using the archive module extension feature of the **mmapplypolicy** command. After completing the parallel execution of the policy-driven archiving process, a collection of image files in this format will remain. These image files are gathered by the **mmimgbackup** command and archived to TSM automatically.

   If using the **-N** *nodes* option, it is recommended that the same operating system be used when running **mmimgbackup**. Note the directory created with **-g** *GlobalWorkDirectory* to store the image files.

8. After the image backup is complete, delete the snapshot used for backup with the following command:

   ```
   mmdelsnapshot smallfs smallfssnap
   ```

Restore procedure with SOBAR

This section provides a detailed example of the restore procedure used with SOBAR.

In order to restore a file system, the configuration data stored from a previous run of **mmbackupconfig** and the image files produced from **mmimgbackup** must be accessible.

Throughout these procedures, the sample file system used is called **smallfs**. Where appropriate, replace this value with your file system name.

1. Restore the metadata image files from **mmimgbackup** and the backup configuration data from **mmbackupconfig** with a **dsmc** command similar to the following:

   ```
   dsmc restore -subdir=yes /u/user/backup/8516/
   ```

2. Retrieve the base file system configuration information.

   Use the **mmrestoreconfig** command to generate a configuration file, which contains the details of the former file system:

   ```
   mmrestoreconfig Device -i InputFile -F QueryResultFile
   ```

3. Recreate NSDs if they are missing.

   Using the output file generated in the previous step as a guide, the administrator might need to recreate NSD devices for use with the restored file system. In the output file, the **NSD configuration** section contains the NSD information; for example:

   ```
   ######## NSD configuration ##################
   ## Disk descriptor format for the mmcrnsd command.
   ## Please edit the disk and desired name fields to match
   ## your current hardware settings.
   ##
   ## The user then can uncomment the descriptor lines and
   ## use this file as input to the -F option.
   #
   # %nsd:
   #   device=DiskName
   #   nsd=nsd8
   #   usage=dataAndMetadata
   #   failureGroup=-1
   #   pool=system
   #
   ```

   If changes are needed, edit the file in a text editor and follow the included instructions to use it as input to the **mmcrnsd** command, then issue the following command:

   ```
   mmcrnsd -F StanzaFile
   ```

4. Recreate the base file system.

   The administrator must recreate the initial file system. The output query file specified in the previous commands can be used as a guide. The following example shows the section of this file that is needed when recreating the file system:

   ```
   ######### File system configuration #############
   ## The user can use the predefined options/option values
   ## when recreating the filesystem.  The option values
   ## represent values from the backed up filesystem.
   #
   # mmcrfs FS_NAME NSD_DISKS -j cluster -k posix -Q yes -L 4194304 --disable-fastea
     -T /fs2 -A no --inode-limit 278016#
   #
   # When preparing the file system for image restore, quota
   # enforcement must be disabled at file system creation time.
   # If this is not done, the image restore process will fail.
   ```

   Do one of the following to recreate the file system:

   - Edit the output file. Uncomment the **mmcrfs** command, and specify the appropriate file system name and NSD disk(s). Remove the **-Q** option to ensure quotas are not enabled. Save the changes and run the file as a shell script:

     ```
     sh OutputFile
     ```

- From the command line, issue an **mmcrfs** command similar to the one in the output file, but specify the appropriate file system name and NSD disk(s). Do not specify the **-Q** option to ensure quotas are not enabled.

5. Restore essential file system configuration.

   Using the **mmrestoreconfig** command, the essential file system configuration can be restored to the file system that was just created in the previous step. Quota is disabled in this step because the quota system must remain inactive until after the file system image has been restored. Filesets will also be restored and linked, if necessary, using a method specific for image restore. The **--image-restore** option should be used to restore the configuration data in the proper format for SOBAR; for example:

   ```
   mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925 --image-restore
   ```
6. Mount the file system in read-only mode for image restore with the following command:

   ```
   mmmount smallfs -o ro
   ```
7. Perform the image restore; for example:

   ```
   mmimgrestore smallfs /u/user/backup/8516/mmPolicy.8551.D4D85229
   ```
8. To optionally display the restored file system structure, use the following command:

   ```
   ls -l /smallfs/*
   ```

   The system displays information similar to the following:

   ```
   -rw-r--r-- 1 root root 1024 Sep 25 11:34 /smallfs/1Kfile.1
   -rw-r--r-- 1 root root 1024 Sep 25 11:34 /smallfs/1Kfile.2
   -rwxr--r-- 1 root root  238 Sep 25 11:34 /smallfs/generateChksums*
   ```
9. Unmount the file system with the following command:

   ```
   mmumount smallfs
   ```
10. Restore quota configuration.

    If any quota enforcement was used in the prior file system, it can be restored now using the **mmrestoreconfig** command. This step will not enable quotas if they were not in use at the time of the configuration backup. To restore the quota configuration, issue a command similar to the following:

    ```
    mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925 -Q only
    ```
11. Mount the file system in read-write mode with the following command:

    ```
    mmmount smallfs
    ```
12. Delete the unusable HSM directory.

    The **.SpaceMan** directory contains file stubs from the former space management control information. This directory must be deleted prior to restarting HSM management. Use the following command:

    ```
    rm -rf /smallfs/.SpaceMan
    ```
13. To optionally restart HSM, use the following command:

    ```
    dsmmigfs restart
    ```
14. Resume HSM management on the newly reconstructed file system, to resume managing disk space and to permit recall of files, with the following command:

    ```
    dsmmigfs add /smallfs
    ```
15. To optionally display the managed file system from HSM, use the following command:

    ```
    dsmls /smallfs/*
    ```

    All files are currently in the migrate state.
16. To optionally begin recalling files by forcing a specific recall, use the following command:

    ```
    dsmrecall -Recursive /smallfs/*
    ```

Chapter 6. Establishing disaster recovery for a cluster

The ability to detect and quickly recover from a massive hardware failure is of paramount importance to businesses that make use of real-time data processing systems.

GPFS provides a number of features that facilitate the implementation of highly-available GPFS environments capable of withstanding catastrophic hardware failures. By maintaining a replica of the file system's data at a geographically-separate location, the system sustains its processing using the secondary replica of the file system in the event of a total failure in the primary environment.

On a high level, a disaster-resilient GPFS cluster is made up of two or three, distinct, geographically-separate hardware sites operating in a coordinated fashion. Two of the sites consist of GPFS nodes and storage resources holding a complete replica of the file system. If a third site is active, it consists of a single node and a single disk used as a tiebreaker for GPFS quorum. In the event of a catastrophic hardware failure that disables the operation of an entire site, and assuming the tiebreaker site remains operational, file system services failover to the remaining subset of the cluster and continue serving the data using the replica of the file system that survived the disaster. However, if the tiebreaker fails during the disaster, the remaining number of nodes and disks is insufficient to satisfy the quorum rules and the surviving site loses access to the GPFS file system. A manual procedure is needed to instruct GPFS to disregard the existing quorum assignments and continue operating with whatever resources are available.

The secondary replica is maintained by one of several methods:

- Synchronous mirroring utilizing GPFS replication.

  The data and metadata replication features of GPFS are used to implement synchronous mirroring between a pair of geographically-separate sites. The use of logical replication-based mirroring offers a generic solution that relies on no specific support from the disk subsystem beyond the basic ability to read and write data blocks. For more information, see the topic *Synchronous mirroring with GPFS replication* in the *IBM Spectrum Scale: Advanced Administration Guide*.
- Synchronous mirroring utilizing IBM TotalStorage Enterprise Storage Server® (ESS) Peer-to-Peer Remote Copy (PPRC).

  The PPRC feature of the ESS establishes a persistent mirroring relationship between pairs of Logical Units (LUNs) on two subsystems connected over an ESCON or a Fibre Channel link. All updates performed on the set of primary, or source, LUNs appear in the same order on the secondary, or target, disks in the target subsystem. The PPRC mechanism provides for an exact bitwise replica of the source's content as seen at the time of the failure on the target should the source volume fail. See *Synchronous mirroring utilizing IBM TotalStorage ESS PPRC*.

  Usage of synchronous IBM TotalStorage Enterprise Storage Server (ESS) Peer-to-Peer Remote Copy (PPRC) now extends to IBM TotalStorage Metro Mirror.

  Usage of asynchronous PPRC now extends to IBM TotalStorage Global Mirror.
- Asynchronous mirroring utilizing ESS FlashCopy®.

  Periodic point-in-time copies of the file system are taken using the facilities of ESS FlashCopy. The copy is subsequently transferred to a remote backup location using PPRC, written to tape, or both. See *Asynchronous mirroring utilizing ESS FlashCopy*.

The primary advantage of both synchronous mirroring methods is the minimization of the risk of permanent data loss. Both methods provide two consistent, up-to-date replicas of the file system, each available for recovery should the other one fail. However, inherent to all solutions that synchronously mirror data over a wide area network link is the latency penalty induced by the replicated write I/Os. This makes both synchronous mirroring methods prohibitively inefficient for certain types of performance-oriented applications. The asynchronous method effectively eliminates this penalty.

However, asynchronous mirroring may result in two distinct and not necessarily consistent replicas of the file system. There is no guarantee as to the validity of data kept in the snapshot beyond the fact that the file system's metadata is consistent and that the user data must have been valid at the time the snapshot was taken.

Synchronous mirroring with GPFS replication

In a configuration utilizing GPFS replication, a single GPFS cluster is defined over three geographically-separate sites consisting of two production sites and a third tiebreaker site. One or more file systems are created, mounted, and accessed concurrently from the two active production sites.

The data and metadata replication features of GPFS are used to maintain a secondary copy of each file system block, relying on the concept of disk failure groups to control the physical placement of the individual copies:

1. Separate the set of available disk volumes into two failure groups. Define one failure group at each of the active production sites.
2. Create a replicated file system. Specify a replication factor of 2 for both data and metadata.

When allocating new file system blocks, GPFS always assigns replicas of the same block to distinct failure groups. This provides a sufficient level of redundancy allowing each site to continue operating independently should the other site fail.

GPFS enforces a node quorum rule to prevent multiple nodes from assuming the role of the file system manager in the event of a network partition. Thus, a majority of quorum nodes must remain active in order for the cluster to sustain normal file system usage. Furthermore, GPFS uses a quorum replication algorithm to maintain the content of the file system descriptor (one of the central elements of the GPFS metadata). When formatting the file system, GPFS assigns some number of disks (usually three) as the descriptor replica holders that are responsible for maintaining an up-to-date copy of the descriptor. Similar to the node quorum requirement, a majority of the replica holder disks must remain available at all times to sustain normal file system operations. This file system descriptor quorum is internally controlled by the GPFS daemon. However, when a disk has failed due to a disaster you must manually inform GPFS that the disk is no longer available and it should be excluded from use.

Considering these quorum constraints, it is suggested that a third site in the configuration fulfill the role of a tiebreaker for the node and the file system descriptor quorum decisions. The tiebreaker site consists of:

1. A single quorum node

   As the function of this node is to serve as a tiebreaker in GPFS quorum decisions, it does not require normal file system access and SAN connectivity. To ignore disk access errors on the tiebreaker node, enable the **unmountOnDiskFail** configuration parameter through the **mmchconfig** command. When enabled, this parameter forces the tiebreaker node to treat the lack of disk connectivity as a local error, resulting in a failure to mount the file system, rather that reporting this condition to the file system manager as a disk failure.
2. A single network shared disk

   The function of this disk is to provide an additional replica of the file system descriptor file needed to sustain quorum should a disaster cripple one of the other descriptor replica disks. Create a network shared disk over the tiebreaker node's internal disk defining:
   - the local node as an NSD server
   - the disk usage as **descOnly**

     The **descOnly** option instructs GPFS to only store file system descriptor information on the disk.

This three-site configuration is resilient to a complete failure of any single hardware site. Should all disk volumes in one of the failure groups become unavailable, GPFS performs a transparent failover to the remaining set of disks and continues serving the data to the surviving subset of nodes with no administrative intervention. While nothing prevents you from placing the tiebreaker resources at one of the active sites, to minimize the risk of double-site failures it is suggested you install the tiebreakers at a third, geographically distinct location.

The high-level organization of a replicated GPFS cluster for synchronous mirroring where all disks are directly attached to all nodes in the cluster is shown in Figure 5. An alternative to this design would be to have the data served through designated NSD servers.

With GPFS release 4.1.0, a new, more fault-tolerant configuration mechanism has been introduced as the successor for the server-based mechanisms. The server-based configuration mechanisms consist of two configuration servers specified as the primary and secondary cluster configuration server. The new configuration mechanism uses all specified quorum nodes in the cluster to hold the GPFS configuration and is called CCR (Clustered Configuration Repository). The CCR is used by default during cluster creation unless the CCR is explicitly disabled. The mmlscluster command reports the configuration mechanism in use in the cluster.

The following sections describe the differences regarding disaster recovery for the two configuration mechanisms.

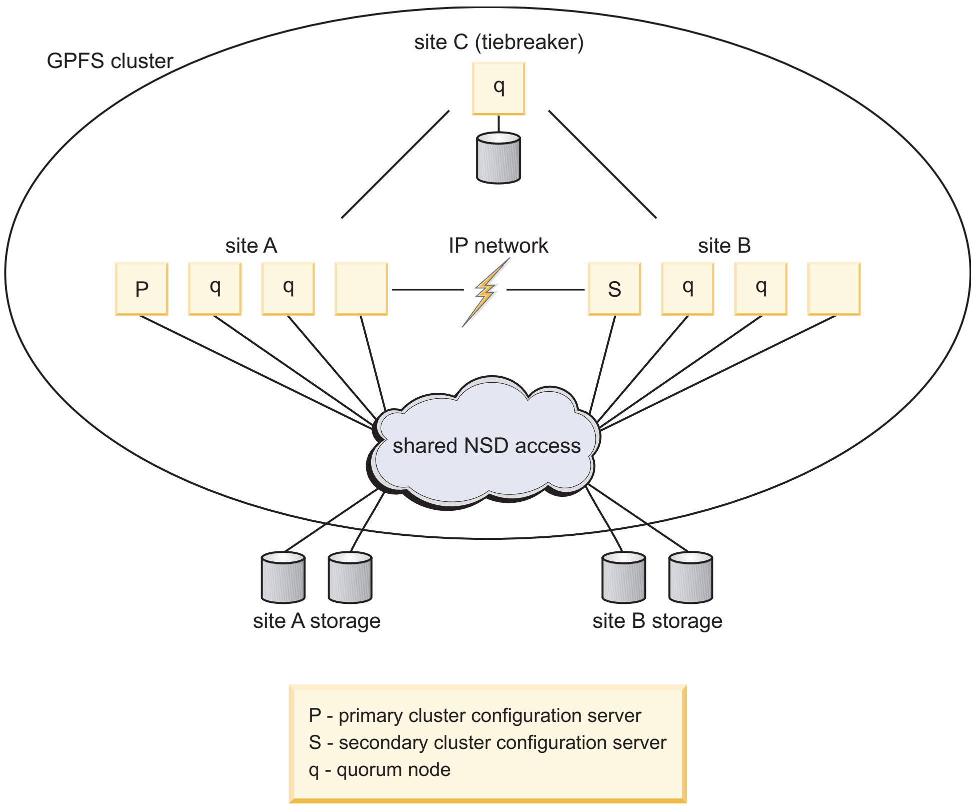


Figure 5. Synchronous mirroring utilizing GPFS replication

Configuring a cluster with GPFS replication

The setting up of a GPFS cluster by utilizing replication can be described through an example.

To establish a disaster-resilient GPFS cluster utilizing replication as shown in Figure 5, consider the configuration:

Site A Consisting of:

- Nodes – **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**

- Disk device names – **diskA1**, **diskA2**

  diskA1 and **diskA2** are SAN-attached and accessible from all nodes at site **A** and site **B**.

Site B Consisting of:
- Nodes – **nodeB001**, **nodeB002**, **nodeB003**, **nodeB004**
- Disks – **diskB1**, **diskB2**

  diskB1 and **diskB2** are SAN-attached and accessible from all nodes at site **A** and site **B**.

Site C (tiebreaker)
Consisting of:
- Node – **nodeC**
- Disk – **diskC**

  diskC is an NSD defined over the internal disk of the node **nodeC** and is directly accessible only from site **C**

1. Create a GPFS cluster selecting **nodeA001** at site A as the primary cluster data server node, **nodeB001** at site **B** as the secondary cluster data server nodes, and the nodes in the cluster contained in the file **clusterNodes**. The **clusterNodes** file contains the node descriptors:
```
nodeA001:quorum-manager
nodeA002:quorum-manager
nodeA003:quorum-manager
nodeA004:client
nodeB001:quorum-manager
nodeB002:quorum-manager
nodeB003:quorum-manager
nodeB004:client
nodeC:quorum-client
```
   Issue this command:
```
mmcrcluster –N clusterNodes
```
   This cluster is created using the new Clustered Configuration Repository (CCR) as its configuration mechanisms. All specified quorum nodes contain the GPFS configuration. The cluster can be created using the traditional, server-based configuration mechanisms by issuing the following command instead:
```
mmcrcluster –N clusterNodes --ccr-disable –p nodeA001 –s nodeB001
```
2. Prevent false disk errors in the SAN configuration from being reported to the file system manager by enabling the **unmountOnDiskFail** option on the tiebreaker node:
```
mmchconfig unmountOnDiskFail=yes -N nodeC
```
3. Define the set of network shared disks for the cluster where disks at sites **A** and **B** are assigned to failure groups 1 and 2, respectively. The tiebreaker disk is assigned to failure group 3. The disk descriptors contained in the file **clusterDisks** are:
```
%nsd: device=/dev/diskA1
    servers=nodeA002,nodeA003
    usage=dataAndMetadata
    failureGroup=1


%nsd: device=/dev/diskA2
    servers=nodeA003,nodeA002
    usage=dataAndMetadata
    failureGroup=1


%nsd: device=/dev/diskB1
    servers=nodeB002,nodeB003
    usage=dataAndMetadata
    failureGroup=2
```

```
%nsd: device=/dev/diskB2
    servers=nodeB003,nodeB002
    usage=dataAndMetadata
    failureGroup=2


%nsd: device=/dev/diskC1
    servers=nodeC
    usage=descOnly
    failureGroup=3
```

Issue this command:

```
mmcrnsd –F clusterDisks
```

4. Issue the **mmlsnsd** command to verify that the network shared disks have been created:

```
mmlsnsd -m
```

Output is similar to this:

```
Disk name    NSD volume ID      Device         Node name         Remarks
---------------------------------------------------------------------------------
 gpfs1nsd     0972445B416BE502   /dev/diskA1    nodeA002          server node
 gpfs1nsd     0972445B416BE502   /dev/diskA1    nodeA003          server node
 gpfs2nsd     0972445B416BE509   /dev/diskA2    nodeA002          server node
 gpfs2nsd     0972445B416BE509   /dev/diskA2    nodeA003          server node
 gpfs3nsd     0972445F416BE4F8   /dev/diskB1    nodeB002          server node
 gpfs3nsd     0972445F416BE4F8   /dev/diskB1    nodeB003          server node
 gpfs4nsd     0972445F416BE4FE   /dev/diskB2    nodeB002          server node
 gpfs4nsd     0972445F416BE4FE   /dev/diskB2    nodeB003          server node
 gpfs5nsd     0972445D416BE504   /dev/diskC1    nodeC             server node
```

5. Start the GPFS daemon on all nodes:

```
mmstartup -a
```

6. Create a replicated file system **fs0**:

```
mmcrfs /gpfs/fs0 fs0 –F clusterDisks –m 2 –M 2 –r 2 –R 2
```

7. Mount **fs0** on all nodes at sites A and B.

Recovery from a disaster with GPFS replication

Utilizing GPFS replication allows for *failover* to the surviving site without disruption of service as long as both the remaining site and the tiebreaker site remain functional. It remains in this state until a decision is made to restore the operation of the affected site by executing the *failback* procedure. If the tiebreaker site is also affected by the disaster and is no longer operational, GPFS quorum is broken and manual intervention is required to resume file system access.

Existing quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements:

1. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to non-quorum nodes. Issue the **mmchnode --nonquorum** command.
2. To relax file system descriptor quorum, temporarily eliminate the failed disks from the group of disks from which the GPFS daemon uses to write the file system descriptor file to. Issue the **mmfsctl exclude** command for each of the failed disks.

While the GPFS cluster is in a failover state, it is suggested that no changes to the GPFS configuration be made. If the server-based configuration mechanism is in use, changes to your GPFS configuration require both cluster configuration servers to be operational. If both servers are not operational, the sites would have distinct, and possibly inconsistent, copies of the GPFS **mmsdrfs** configuration data file. While the servers can be migrated to the surviving site, it is best to avoid this step if the disaster does not leave the affected site permanently disabled.

If it becomes absolutely necessary to modify the GPFS configuration while in failover mode, for example to relax quorum, you must ensure that all nodes at the affected site are powered down and left in a stable inactive state. They must remain in such state until the decision is made to execute the failback procedure. As a means of precaution, we suggest disabling the GPFS autoload option on all nodes to prevent GPFS from bringing itself up automatically on the affected nodes should they come up spontaneously at some point after a disaster.

Failover to the surviving site

Following a disaster, which failover process is implemented depends upon whether or not the tiebreaker site is affected.

Failover without the loss of tiebreaker site C

The proposed three-site configuration is resilient to a complete failure of any single hardware site. Should all disk volumes in one of the failure groups become unavailable, GPFS performs a transparent failover to the remaining set of disks and continues serving the data to the surviving subset of nodes with no administrative intervention.

Failover with the loss of tiebreaker site C with server-based configuration in use

If both site **A** and site **C** fail:

1. Shut the GPFS daemon down on the surviving nodes at site **B**, where the file **gpfs.siteB** lists all of the nodes at site **B**:
   ```
   mmshutdown -N gpfs.siteB
   ```
2. If it is necessary to make changes to the configuration, migrate the primary cluster configuration server to a node at site **B**:
   ```
   mmchcluster -p nodeB002
   ```
3. Relax node quorum by temporarily changing the designation of each of the failed quorum nodes to non-quorum nodes:
   ```
   mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC
   ```
4. Relax file system descriptor quorum by informing the GPFS daemon to migrate the file system descriptor off of the failed disks:
   ```
   mmfsctl fs0 exclude -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   ```
5. Restart the GPFS daemon on the surviving nodes:
   ```
   mmstartup -N gpfs.siteB
   ```
6. Mount the file system on the surviving nodes at site B.

Failover with the loss of tiebreaker site C with Clustered Configuration Repository (CCR) in use

If both site A and site C fail:

1. Shut the GPFS daemon down on the surviving nodes at site B , where the file gpfs.siteB lists all of the nodes at site B :
   ```
   mmdsh -N gpfs.siteB /usr/lpp/mmfs/bin/mmshutdown
   ```
2. Changing (downgrading) the quorum assignments when half or more of the quorum nodes are no longer available at site B using the `–– force` option :
   ```
   mmchnode ––nonquorum -N nodeA001,nodeA002,nodeA003,nodeC --force
   ```
3. Relax file system descriptor quorum by informing the GPFS daemon to migrate the file system descriptor off of the failed disks:
   ```
   mmfsctl fs0 exclude -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   ```
4. Restart the GPFS daemon on the surviving nodes:
   ```
   mmstartup -N gpfs.siteB
   ```

5. Mount the file system on the surviving nodes at site B.

Make no further changes to the quorum designations at site B until the failed sites are back on line and the following failback procedure has been completed.

Do not shut down the current set of nodes on the surviving site B and restart operations on the failed sites A and C. This will result in a non-working cluster.

Failback procedures

Which failback procedure you follow depends upon whether the nodes and disks at the affected site have been repaired or replaced.

If the disks have been repaired, you must also consider the state of the data on the failed disks:
- For nodes and disks that have been repaired and *you are certain* the data on the failed disks has not been changed, follow either:
  - *failback with temporary loss and no configuration changes*
  - *failback with temporary loss and configuration changes*
- If the nodes have been replaced and either the disks have been replaced or repaired, and *you are not certain* the data on the fail disks has not been changed, follow the procedure for *failback with permanent loss*.

Delayed failures: In certain failure cases the loss of data may not be immediately apparent. For example, consider this sequence of events:
1. Site **B** loses connectivity with sites **A** and **C**.
2. Site **B** then goes down due to loss of node quorum.
3. Sites **A** and **C** remain operational long enough to modify some of the data on disk but suffer a disastrous failure shortly afterwards.
4. Node and file system descriptor quorums are overridden to enable access at site **B**.

Now the two replicas of the file system are inconsistent and the only way to reconcile these copies during recovery is to:
1. Remove the damaged disks at sites **A** and **C**.
2. Either replace the disk and format a new NSD or simply reformat the existing disk if possible.
3. Add the disk back to the file system, performing a full resynchronization of the file system's data and metadata and restore the replica balance using the **mmrestripefs** command.

Failback with temporary loss and no configuration changes:

If the outage was of a temporary nature and your configuration has not been altered, it is a simple process to fail back to the original state.

After all affected nodes and disks have been repaired and *you are certain* the data on the failed disks has not been changed:
1. Start GPFS on the repaired nodes where the file **gpfs.sitesAC** lists all of the nodes at sites **A** and **C**:
   ```
   mmstartup -N gpfs.sitesAC
   ```
2. Restart the affected disks. If more than one disk in the file system is down, they must all be started at the same time:
   ```
   mmchdisk fs0 start -a
   ```

Failback with temporary loss and configuration changes in the server-based configuration:

If the outage was of a temporary nature and your configuration has been altered, follow this procedure to fail back to the original state in case primary and secondary configuration servers are in use.

After all affected nodes and disks have been repaired and *you are certain* the data on the failed disks has not been changed:

1. Ensure that all nodes have the latest copy of the **mmsdrfs** file:
   ```
   mmchcluster -p LATEST
   ```
2. Migrate the primary cluster configuration server back to site **A**:
   ```
   mmchcluster -p nodeA001
   ```
3. Restore node quorum designations at sites **A** and **C**:
   ```
   mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
   ```
4. Start GPFS on the repaired nodes where the file **gpfs.sitesAC** lists all of the nodes at sites **A** and **C**:
   ```
   mmstartup -N gpfs.sitesAC
   ```
5. Restore the file system descriptor quorum by informing the GPFS to include the repaired disks:
   ```
   mmfsctl fs0 include -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   ```
6. Bring the disks online and restripe the file system across all disks in the cluster to restore the initial replication properties:
   ```
   mmchdisk fs0 start -a
   mmrestripefs fs0 -b
   ```

   The **-r** flag may be used on the **mmrestripefs** command instead.

Failback with temporary loss using the Clustered Configuration Repository (CCR) configuration mechanism:

If the outage was of a temporary nature and your configuration has been altered, follow this procedure to failback to the original state, in case the Clustered Configuration Repository (CCR) configuration scheme is in use.

After all affected nodes and disks have been repaired and you are certain the data on the failed disks has not been changed, complete the following steps.

1. Shut down the GPFS daemon on the surviving nodes at site B , and on the former failed and now recovered sites A and C , where the file `gpfs.siteB` lists all of the nodes at site B and the file `gpfs.siteA` lists all of the nodes at site A and the tiebreaker node at site C:
   ```
   mmshutdown -N gpfs.siteB
   mmshutdown -N gpfs.siteA
   mmshutdown -N nodeC
   ```
2. Restore original node quorum designation for the tiebreaker site C at site B and start GPFS on site C:
   ```
   mmstartup -N gpfs.siteB
   mmchnode --quorum -N nodeC
   mmstartup -N nodeC
   ```
3. Restore original node quorum designation for site A at site B and start GPFS on site A:
   ```
   mmchnode --quorum -N nodeA001,nodeA002,nodeA003
   mmstartup -N gpfs.siteA
   ```
4. Restore the file system descriptor quorum by informing the GPFS to include the repaired disks:
   ```
   mmumount fs0 -a;mmfsctl fs0 include -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   ```
5. Mount the file system on all nodes at sites A and B.

   Note: Do not allow the failed sites A and C to come online at the same time or when site B is unavailable or not functional.
6. Bring the disks online and restripe the file system across all disks in the cluster to restore the initial replication properties:
   ```
   mmchdisk fs0 start -a
   mmrestripefs fs0 -b
   ```

The `-r` flag can be used on the `mmrestripefs` command instead.

Failback with permanent loss:

If an outage is of a permanent nature, follow steps to remove and replace the failed resources, and then resume the operation of GPFS across the cluster.

1. Remove the failed resources from the GPFS configuration
2. Replace the failed resources, then add the new resources into the configuration
3. Resume the operation of GPFS across the entire cluster

Assume that sites **A** and **C** have had permanent losses. To remove all references of the failed nodes and disks from the GPFS configuration and replace them:

Procedure when the server-based configuration scheme is in use

1. To remove the failed resources from the GPFS configuration:
   a. If as part of the failover process, *you did not* migrate the primary cluster configuration server, migrate the server to node **nodeB002** at site B:
   ```
   mmchcluster –p nodeB002
   ```
   b. Delete the failed disks from the GPFS configuration:
   ```
   mmdeldisk fs0 "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   mmdelnsd "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   ```
   c. Delete the failed nodes from the GPFS configuration:
   ```
   mmdelnode -N nodeA001,nodeA002,nodeA003,nodeA004,nodeC
   ```
2. If there are new resources to add to the configuration:
   a. Add the new nodes at sites **A** and **C** to the cluster where the file **gpfs.sitesAC** lists of the new nodes:
   ```
   mmaddnode -N gpfs.sitesAC
   ```
   b. Ensure that all nodes have the latest copy of the **mmsdrfs** file:
   ```
   mmchcluster -p LATEST
   ```
   c. Migrate the primary cluster configuration server back to site **A**:
   ```
   mmchcluster -p nodeA001
   ```
   d. Start GPFS on the new nodes
   ```
   mmstartup -N gpfs.sitesAC
   ```
   e. Prepare the new disks for use in the cluster, create the NSDs using the original disk descriptors for site **A** contained in the file **clusterDisksAC**:
   ```
   %nsd: device=/dev/diskA1
       servers=nodeA002,nodeA003
       usage=dataAndMetadata
       failureGroup=1

   %nsd: device=/dev/diskA2
       servers=nodeA003,nodeA002
       usage=dataAndMetadata
       failureGroup=1

   %nsd: device=/dev/diskC1
       servers=nodeC
       usage=descOnly

   failureGroup=3mmcrnsd -F clusterDisksAC
   ```
   f. Add the new NSDs to the file system specifying the **-r** option to rebalance the data on all disks:
   ```
   mmadddisk fs0 -F clusterDisksAC -r
   ```

Procedure when Clustered Configuration Repository (CCR) is in use

1. To remove the failed resources from the GPFS configuration:
   a. Delete the failed disks from the GPFS configuration:
      ```
      mmdeldisk fs0 "gpfs1nsd;gpfs2nsd;gpfs5nsd"
      ```
      ```
      mmdelnsd "gpfs1nsd;gpfs2nsd;gpfs5nsd"
      ```
   b. Delete the failed nodes from the GPFS configuration:
      ```
      mmdelnode -N nodeA001,nodeA002,nodeA003,nodeA004,nodeC
      ```
2. If there are new resources to add to the configuration:
   a. Add the new nodes at sites **A** and **C** to the cluster where the file **gpfs.sitesAC** lists of the new nodes:
      ```
      mmaddnode -N gpfs.sitesAC
      ```
   b. Restore original quorum node assignments at site B:
      ```
      mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
      ```
   c. Start GPFS on the new nodes
      ```
      mmstartup -N gpfs.sitesAC
      ```
   d. Prepare the new disks for use in the cluster, create the NSDs using the original disk descriptors for site **A** contained in the file **clusterDisksAC**:
      ```
      %nsd: device=/dev/diskA1
       servers=nodeA002,nodeA003
       usage=dataAndMetadata
       failureGroup=1

       %nsd: device=/dev/diskA2
       servers=nodeA003,nodeA002
       usage=dataAndMetadata
       failureGroup=1

       %nsd: device=/dev/diskC1
       servers=nodeC
       usage=descOnly

       failureGroup=3mmcrnsd -F clusterDisksAC
      ```
   e. Add the new NSDs to the file system specifying the **-r** option to rebalance the data on all disks:
      ```
      mmadddisk fs0 -F clusterDisksAC -r
      ```

Synchronous mirroring with IBM TotalStorage ESS PPRC

PPRC is a function that continuously updates a secondary (target) copy of an ESS disk volume to match changes made to a primary (source) volume. Any pair of equal-sized ESS disks can be configured for a PPRC relationship, during which all write operations performed on the source are synchronously mirrored to the target device.

The PPRC protocol guarantees that the secondary copy is constantly up-to-date by ensuring that the primary copy is written only if the primary storage subsystem received acknowledgment that the secondary copy has been written. The paired volumes typically reside on two distinct and geographically separated ESS devices communicating over ESCON or over a Fibre Channel link.

A number of PPRC tasks are provided to facilitate recovery in the event of a site-wide failure. After the failure of the primary volume (or the failure of the entire storage subsystem), users execute the PPRC failover task, which suspends the PPRC relationship between the given pair of volumes and turns the target volume into a primary. When a volume enters the suspended state, a modification bitmap is established to keep track of the write operations performed on that volume to allow for an efficient resynchronization.

Once the operation of the original primary volume has been restored, the PPRC failback task is executed to resynchronize the content of the two volumes. The original source volume is switched to the target mode, after which all modified data tracks (those recorded in the modification bitmap) are copied from the original target disk. The volume pair is then suspended again and another task is executed to reverse the volumes' roles, thus bringing the pair into its initial state.

The ESS Copy Services Web Interface User's Guide as described in the *IBM TotalStorage Enterprise Storage Server Web Interface User's Guide*, is a GUI that allows users to establish and terminate PPRC pairs, and invoke failover, failback, and related PPRC functions. A Java-based command-line interface as described in the *IBM Enterprise Storage Server Command-Line Interfaces User's Guide*, provides another method of interaction.

A PPRC-based GPFS cluster can be established in two manners:

- A single GPFS cluster encompassing two sites and an optional tiebreaker site
- Two distinct GPFS clusters

An active-active cluster

In an active-active cluster, a single GPFS cluster contains two active sites and an optional tiebreaker site.

The high-level organization of PPRC-based active-active GPFS cluster is illustrated in Figure 6 on page 94. A single GPFS cluster is created over three sites. The data is mirrored between two active sites with a cluster configuration server residing at each site and a tiebreaker quorum node installed at the third location. The presence of an optional tiebreaker node allows the surviving site to satisfy the node quorum requirement with no additional intervention. Without the tiebreaker, the failover procedure requires an additional administrative command to relax node quorum and allow the remaining site to function independently. Furthermore, the nodes at the recovery site have direct disk paths to the primary site's storage.

The GPFS configuration resides either on the two configuration server (primary and secondary), when the cluster has been created with the Clustered Configuration Repository (CCR) disable option (mmcrcluster), or on each quorum node, when the cluster has Clustered Configuration Repository (CCR) enabled, or on the primary/secondary, when the Clustered Configuration Repository (CCR) is disabled.

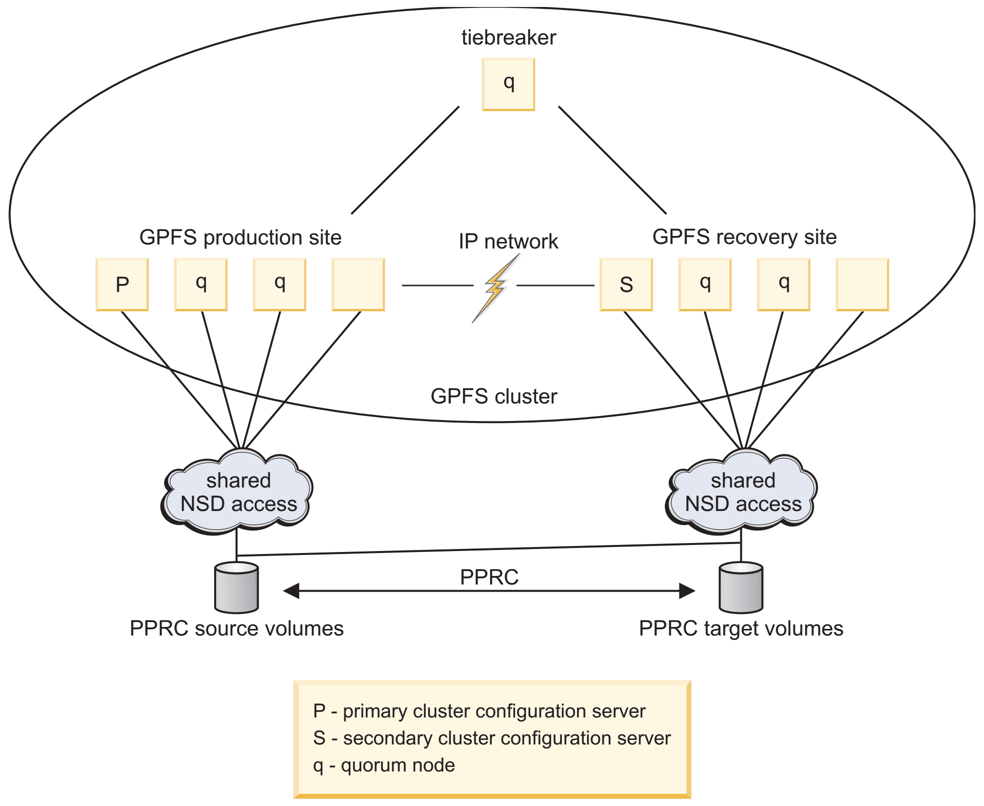


Figure 6. A synchronous active-active PPRC-based mirrored GPFS configuration with a tiebreaker site

Setting up an active-active GPFS configuration

This example demonstrates how to configure an active-active GPFS cluster.

To establish an active-active PPRC-based GPFS cluster with a tiebreaker site as shown in Figure 6, consider the configuration:

Site A (production site)

Consists of:

- Nodes – **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**
- Storage subsystems – Enterprise Storage Server (ESS) **A**, logical subsystem (LSS) **A**
- Disk volumes – **diskA** on LSS **A**

  diskA is SAN-attached and accessible from sites **A** and **B**

Site B (recovery site)

Consists of:

- Nodes – **nodeB001**, **nodeB002**, **nodeB003**, **nodeB004**
- Storage subsystems – ESS **B**, LSS **B**
- Disk volumes – **diskB** on LSS **B**

  diskB is SAN-attached and accessible from site **B** only

Site C (tiebreaker)

Consists of:

- Nodes – **nodeC**

  diskC is an NSD defined over the internal disk of the node **nodeC** and is directly accessible only from site **C**

1. Create a dual-active ESS copy services domain assigning ESS A as **ServerA** and ESS B as **ServerB**. See the *IBM Enterprise Storage Server User's Guide*. In order to provide for the availability of at least one server after a disaster, the servers should reside at different sites.
2. Establish a PPRC logical path from LSS **A** to LSS **B**. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
3. In order to protect the order of dependent writes that span multiple disk volumes, multiple LSS devices, or both, a consistency group should be defined over all logical subsystems at the primary site. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*. In this case that would only be LSS A. See *Data integrity and the use of PPRC consistency groups*.
4. Establish a synchronous PPRC volume pair between the source and target using the **copy entire volume** option and leave the **permit read from secondary** option disabled. In this case it would be diskA-diskB. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
5. Create a GPFS cluster defining the primary cluster configuration server as nodes **nodeA001** at site **A**, the secondary cluster configuration server as **nodeB001** at site **B**, an equal number of quorum nodes at each site, including the tiebreaker node at site **C**, **nodeC**. To prevent the tiebreaker node from assuming the role of file system manager, define it as **client**. Define all other quorum nodes as **manager**. List the nodes in the cluster in the file **NodeDescFile**. The **NodeDescFile** file contains the node descriptors:
   ```
   nodeA001:quourum-manager
   nodeA002:quorum-manager
   nodeA003:quorum-manager
   nodeA004:client
   nodeB001:quorum-manager
   nodeB002:quorum-manager
   nodeB003:quorum-manager
   nodeB004:client
   nodeC:quorum-client
   ```
   Issue this command:
   ```
   mmcrcluster –N NodeDescFile –p nodeA001 –s nodeB001
   ```
6. Enable the **unmountOnDiskFail** option on the tiebreaker node preventing false disk errors in the SAN configuration from being reported to the file system manager by issuing the **mmchconfig** command:
   ```
   mmchconfig unmountOnDiskFail=yes -N nodeC
   ```
7. Create an NSD over **diskA**. The disk descriptor contained in the file **DiskDescFile** is:
   ```
   /dev/diskA:nodeA001:nodeA002:dataAndMetadata:1
   ```
   Issue this command:
   ```
   mmcrnsd –F DiskDescFileP
   ```
8. Start the GPFS daemon on all nodes:
   ```
   mmstartup -a
   ```
9. Create a GPFS file system and mount it on all nodes at sites **A** and **B**.
   ```
   mmcrfs /gpfs/fs0 fs0 -F DiskDescFile
   ```

Failover and failback for an active-active configuration

For an active-active PPRC-based cluster, complete these steps to restore access to the file system through site **B** after site **A** has experienced a disastrous failure.

Procedure when the server-based configuration scheme is in use

1. Stop the GPFS daemon on the surviving nodes as site **B** where the file **gpfs.siteB** lists all of the nodes at site **B**:
   ```
   mmdsh -N gpfs.siteB /usr/lpp/mmfs/bin/mmshutdown
   ```

2. Perform PPRC failover, establishing a synchronous PPRC pair **diskB**-**diskA** with the PPRC failover option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*. The PPRC state of **diskB** is changed from *duplex target* to *suspended source* and is available for regular I/O. Refer to the *IBM Enterprise Storage Server* for a detailed explanation of PPRC failover.
3. If you needed to relax node quorum or make configuration changes, migrate the primary cluster configuration server to site **B**, issue this command:
   ```
   mmchcluster -p nodeB001
   ```
4. If site **C**, the tiebreaker, failed along with site **A**, existing node quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to non-quorum nodes:
   ```
   mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC
   ```
5. Ensure the source volumes are *not* accessible to the recovery site:
   - Disconnect the cable
   - Define the **nsddevices** user exit file to exclude the source volumes
6. Restart the GPFS daemon on all surviving nodes:
   ```
   mmstartup -N gpfs.siteB
   ```

Procedure when the Clustered Configuration Repository (CCR) scheme is in use

For an active-active PPRC-based cluster, follow these steps to restore access to the file system through site **B** after site **A** has experienced a disastrous failure:

1. Stop the GPFS daemon on the surviving nodes as site **B** where the file gpfs.siteB lists all of the nodes at site **B**:
   ```
   mmshutdown -N gpfs.siteB
   ```
2. Perform PPRC failover, establishing a synchronous PPRC pair **diskB - diskA** with the PPRC failover option. See *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments* . The PPRC state of **diskB** is changed from *duplex target* to *suspended source* and is available for regular I/O. Refer to *IBM Enterprise Storage Server* for a detailed explanation of PPRC failover.
3. If site **C**, the tiebreaker, failed along with site **A**, existing node quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to nonquorum nodes using the `-- force` option:
   ```
   mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC --force
   ```
4. Ensure that the source volumes are not accessible to the recovery site:
   - Disconnect the cable.
   - Define the **nsddevices** user exit file to exclude the source volumes.
5. Restart the GPFS daemon on all surviving nodes:
   ```
   mmstartup -N gpfs.siteB
   ```

Note:
- Make no further changes to the quorum designations at site B until the failed sites are back on line and the following failback procedure has been completed.
- Do not shut down the current set of nodes on the surviving site B and restart operations on the failed sites A and C. This will result in a non-working cluster.

Failback procedure

After the operation of site **A** has been restored, the failback procedure is completed to restore the access to the file system from that location. The following procedure is the same for both configuration schemes (server-based and Clustered Configuration Repository (CCR)). The failback operation is a two-step process:

1. Resynchronize the paired volumes by establishing a temporary PPRC pair with **diskB** defined as the source and **diskA** as the target. The modification bitmap is traversed to find the mismatching disk sectors, whose content is then copied from **diskB** to **diskA**.
   a. Establish a PPRC path from LSS **B** to LSS **A** enabling the **consistency group** option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   b. Establish a synchronous PPRC pair **diskB-diskA** with the PPRC **failback** option and **permit read from secondary** left disabled. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   c. Wait for the volume pair to reach the *duplex* (fully synchronized) state.
2. Shut GPFS down at site **B** and reverse the disk roles (the original primary disk becomes the primary again), bringing the PPRC pair to its initial state.
   a. Stop the GPFS daemon on all nodes.
   b. Establish a synchronous PPRC pair **diskA-diskB** with the PPRC failover option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   c. Establish a synchronous PPRC pair **diskA-diskB** with the PPRC failback option and **permit read from secondary** left disabled. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   d. If during failover you migrated the primary cluster configuration server to a node in site **B**:
      1) Migrate the primary cluster configuration server back to site **A**:
         ```
         mmchcluster -p nodeA001
         ```
      2) Restore the initial quorum assignments:
         ```
         mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
         ```
      3) Ensure that all nodes have the latest copy of the **mmsdrfs** file:
         ```
         mmchcluster -p LATEST
         ```
   e. Ensure the source volumes *are* accessible to the recovery site:
      - Reconnect the cable
      - Edit the **nsddevices** user exit file to *include* the source volumes
   f. Start the GPFS daemon on all nodes:
      ```
      mmstartup -a
      ```
   g. Mount the file system on all the nodes at sites **A** and **B**.

An active-passive cluster

In an active-passive environment, two GPFS clusters are set up in two geographically distinct locations (the production and the recovery sites). These clusters are referred to as peer GPFS clusters.

A GPFS file system is defined over a set of disk volumes located at the production site and these disks are mirrored using PPRC to a secondary set of volumes located at the recovery site. During normal operation, only the nodes in the production GPFS cluster mount and access the GPFS file system at any given time, which is the primary difference between a configuration of this type and the active-active model.

In the event of a catastrophe in the production cluster, the PPRC failover task is executed to enable access to the secondary replica of the file system located on the target PPRC disks. See IBM Redbooks® for *IBM TotalStorage Enterprise Storage Server Implementing ESS Copy Services in Open Environments* for a detailed explanation of PPRC failover and failback and the means of invoking these procedures in your environment.

The secondary replica is then mounted on nodes in the recovery cluster as a regular GPFS file system, thus allowing the processing of data to resume at the recovery site. At a latter point, after restoring the physical operation of the production site, we execute the failback procedure to resynchronize the content of the PPRC volume pairs between the two clusters and re-enable access to the file system in the production environment.

The high-level organization of synchronous active-passive PPRC-based GPFS cluster is shown in Figure 7.

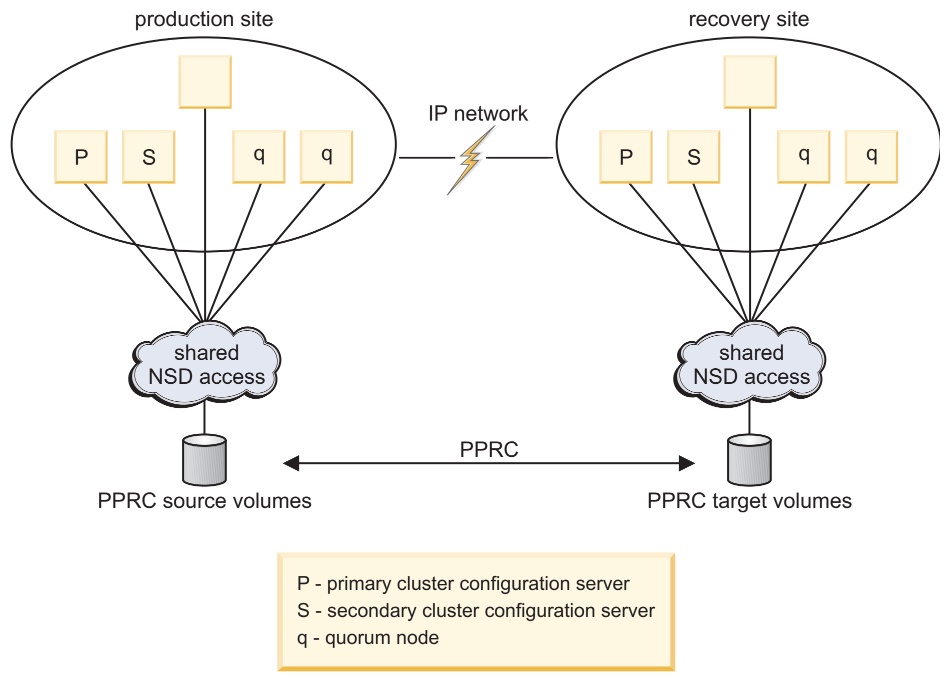


Figure 7. A synchronous active-passive PPRC-based GPFS configuration without a tiebreaker site

Setting up an active-passive GPFS configuration

This example demonstrates how to configure an active-passive GPFS cluster.

To establish an active-passive PPRC based GPFS cluster using hardware replication with tiebreaker site as shown in Figure 7, consider the configuration:

Production site

Consists of:

- Nodes – **nodeP001**, **nodeP002**, **nodeP003**, **nodeP004**, **nodeP005**
- Storage subsystems – Enterprise Storage Server (ESS) **P**, logical subsystem (LSS) **P**
- LUN ids and disk volume names – **lunP1** (**hdisk11**), **lunP2** (**hdisk12**), **lunP3** (**hdisk13**), **lunP4** (**hdisk14**)

Recovery site

Consists of:

- Nodes – **nodeR001**, **nodeR002**, **nodeR003**, **nodeR004**, **nodeR005**
- Storage subsystems – ESS **R**, LSS **R**
- LUN ids and disk volume names – **lunR1** (**hdisk11**), **lunR2** (**hdisk12**), **lunR3** (**hdisk13**), **lunR4** (**hdisk14**)

All disks are SAN-attached and directly accessible from all local nodes.

1. Create a dual-active ESS copy services domain assigning ESS **P** as **ServerA** and ESS **R** as **ServerB**. See the *IBM Enterprise Storage Server User's Guide*.

2. Establish a PPRC logical path from LSS **P** to LSS **R** enabling the consistency group option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*, section on *Data integrity and the use of PPRC consistency groups*.
3. Establish synchronous PPRC volume pairs using the **copy entire volume** option and leave the **permit read from secondary** option disabled:

   ```
   lunP1-lunR1 (source-target)
   lunP2-lunR2 (source-target)
   lunP3-lunR3 (source-target)
   lunP4-lunR4 (source-target)
   ```

   See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
4. Create the recovery cluster selecting **nodeR001** as the primary cluster data server node, **nodeR002** as the secondary cluster data server nodes, and the nodes in the cluster contained in the file **NodeDescFileR**. The **NodeDescFileR** file contains the node descriptors:

   ```
   nodeR001:quourum-manager
   nodeR002:quourum-manager
   nodeR003:quourum-manager
   nodeR004:quourum-manager
   nodeR005
   ```

   Issue this command:

   ```
   mmcrcluster –N NodeDescFileR –p nodeR001 –s nodeR002
   ```
5. Create the GPFS production cluster selecting **nodeP001** as the primary cluster data server node, **nodeP002** as the secondary cluster data server node, and the nodes in the cluster contained in the file **NodeDescFileP**. The **NodeDescFileP** file contains the node descriptors:

   ```
   nodeP001:quourum-manager
   nodeP002:quourum-manager
   nodeP003:quourum-manager
   nodeP004:quourum-manager
   nodeP005
   ```

   Issue this command:

   ```
   mmcrcluster –N NodeDescFileP –p nodeP001 –s nodeP002
   ```
6. At all times the peer clusters must see a consistent image of the mirrored file system's configuration state contained in the **mmsdrfs** file. After the initial creation of the file system, all subsequent updates to the local configuration data must be propagated and imported into the peer cluster. Execute the **mmfsctl syncFSconfig** command to resynchronize the configuration state between the peer clusters after each of these actions in the primary GPFS cluster:
   - Addition of disks through the **mmadddisk** command
   - Removal of disks through the **mmdeldisk** command
   - Replacement of disks through the **mmrpldisk** command
   - Modifications to disk attributes through the **mmchdisk** command
   - Changes to the file system's mount point through the **mmchfs -T** command

   To automate the propagation of the configuration state to the recovery cluster, activate and use the **syncFSconfig** user exit. Follow the instructions in the prolog of **/usr/lpp/mmfs/samples/ syncfsconfig.sample**.
7. From a node in the production cluster, start the GPFS daemon on all nodes:

   ```
   mmstartup -a
   ```
8. Create the NSDs at the production site. The disk descriptors contained in the file **DiskDescFileP** are:

   ```
   /dev/hdisk11:nodeP001:nodeP002:dataAndMetadata:-1
   /dev/hdisk12:nodeP001:nodeP002:dataAndMetadata:-1
   /dev/hdisk13:nodeP001:nodeP002:dataAndMetadata:-1
   /dev/hdisk14:nodeP001:nodeP002:dataAndMetadata:-1
   ```

Issue this command:

```
mmcrnsd –F DiskDescFileP
```

9. Create the GPFS file system and mount it on all nodes at the production site:

```
mmcrfs /gpfs/fs0 fs0 -F DiskDescFileP
```

Failover to the recovery site and subsequent failback for an active-passive configuration

For an active-passive PPRC-based cluster, complete these steps to fail over production to the recovery site.

Procedure when the server-based configuration scheme is in use

1. If the GPFS daemon is not already stopped on all surviving nodes in the production cluster, from a node in the production cluster issue:

```
mmshutdown –a
```

2. Perform PPRC failover. This step involves establishing synchronous PPRC pairs with the failover option:
   - **lunR1-lunP1**
   - **lunR2-lunP2**
   - **lunR3-lunP3**
   - **lunR4-lunP4**

   See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*. After the completion of this step, the volumes on ESS R enter the **suspended source** PPRC state.
3. From a node in the recovery cluster start GPFS:

```
mmstartup –a
```

4. Mount the file system on all nodes in the recovery cluster.

Procedure when the Clustered Configuration Repository (CCR) scheme is in use

1. Stop the GPFS daemon on the surviving nodes as site **B** where the file `gpfs.siteB` lists all of the nodes at site **B**:

```
mmshutdown -N gpfs.siteB
```

2. Perform PPRC failover, establishing a synchronous PPRC pair **diskB - diskA** with the PPRC failover option. See *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*. The PPRC state of **diskB** is changed from duplex target to suspended source and is available for regular I/O. Refer to *IBM Enterprise Storage Server* for a detailed explanation of PPRC failover.
3. If site **C**, the tiebreaker, failed along with site **A**, existing node quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to nonquorum nodes using the `-- force` option:

```
mmchnode --nonquorum -N nodeA001, nodeA002, nodeA003, nodeC --force
```

4. Ensure that the source volumes are *not* accessible to the recovery site:
   - Disconnect the cable
   - Define the **nsddevices** user exit file to exclude the source volumes
5. Restart the GPFS daemon on all surviving nodes:

```
mmstartup -N gpfs.siteB
```

Note: Make no further changes to the quorum designations at site B until the failed sites are back on line and the following failback procedure has been completed. Do not shut down the current set of nodes on the surviving site B and restart operations on the failed sites A and C. This will result in a non-working cluster.

Failback procedure

After the physical operation of the production site has been restored, complete the failback procedure to transfer the file system activity back to the production GPFS cluster. The following procedure is the same for both configuration schemes (server-based and Clustered Configuration Repository (CCR)) The failback operation is a two-step process:

1. For each of the paired volumes, resynchronize the pairs by establishing a temporary PPRC pair with the recovery LUN acting as the sources for the production LUN. The PPRC modification bitmap is traversed to find the mismatching disk tracks, whose content is then copied from the recovery LUN to the production LUN.
   a. Establish a PPRC path from LSS **R** to LSS **P** enabling the **consistency group** option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   b. Establish synchronous PPRC volume pairs with the **failback** option, leaving the **permit read from secondary** option disabled.
      - **lunR1-lunP1**
      - **lunR2-lunP2**
      - **lunR3-lunP3**
      - **lunR4-lunP4**

      See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   c. Wait for all PPRC pairs to reach the **duplex** (fully synchronized) state.
   d. If the state of the system configuration has changed, update the GPFS configuration data in the production cluster to propagate the changes made while in failover mode. From a node at the recovery site, issue:
      ```
      mmfsctl all syncFSconfig –n gpfs.sitePnodes
      ```
2. Stop GPFS on all nodes in the recovery cluster and reverse the disk roles so the original primary disks become the primaries again:
   a. From a node in the recovery cluster, stop the GPFS daemon on all nodes in the recovery cluster:
      ```
      mmshutdown –a
      ```
   b. Establish synchronous PPRC volume pairs with the PPRC failover option:
      - **lunP1-lunR1**
      - **lunP2-lunR2**
      - **lunP3-lunR3**
      - **lunP4-lunR4**

      See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   c. Establish synchronous PPRC volume pairs with the PPRC failback option, leaving the permit **read from secondary** option disabled:
      - **lunP1-lunR1**
      - **lunP2-lunR2**
      - **lunP3-lunR3**
      - **lunP4-lunR4**

      See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   d. From a node in the production cluster, start GPFS:
      ```
      mmstartup –a
      ```
   e. From a node in the production cluster, mount the file system on all nodes in the production cluster.

Data integrity and the use of PPRC consistency groups

The integrity of the post-disaster replica of the file system that is contained on the secondary PPRC disk volumes depends on the assumption that the order of dependent write operations is preserved by the PPRC mirroring mechanism. While the synchronous nature of PPRC guarantees such ordering during periods of stability, certain types of rolling failure scenarios require additional consideration.

By default, without the PPRC **consistency group** option enabled, if the primary ESS detects the loss of the physical PPRC connectivity or the failure of one of the secondary disks, it responds by moving the corresponding primary volumes to the suspended state but continues to process the subsequent I/O requests as normal. The subsystem does not report this condition to the driver and, as a result, the application continues normal processing without any knowledge of the lost updates. GPFS relies on log recovery techniques to provide for the atomicity of updates to the file system's metadata, which is why such behavior would expose GPFS to a serious data integrity risk. Therefore to provide for the proper ordering of updates to the recovery copy of the file system, it is suggested you always enable the **consistency group** option when configuring the PPRC paths between the peer logical subsystems.

Figure 8 on page 103 illustrates this problem. Consider a setup with two primary and two secondary subsystems, each providing access to two LUNs. The four primary LUNs make up the primary replica of the file system. At some point, GPFS attempts to issue a sequence of four write requests, one to each primary disk, with the expectation that the updates appear in the exact order they were issued. If PPRC path 1 breaks before the start of the first write, the recovery site receives updates 3 and 4, but not necessarily 1 and 2 – a result that violates the write dependency rule and renders the target replica of the file system unusable.

The PPRC **consistency group** option determines the behavior of the primary subsystem in situations where a sudden failure of the secondary volume, or a failure of the inter-site interconnect, makes it impossible to sustain the normal synchronous mirroring process. If the PPRC path between the pair has been defined with the **consistency group** option, the primary volume enters a long busy state, and the subsystem reports this condition back to the host system with the QUEUE FULL (QF) SCSI status byte code. Typically, the driver makes several attempts to re-queue the request and ultimately reports the failure to the requestor. This allows GPFS to execute the appropriate action in response to the failure by either marking the disk down or panicking the file system. See the *IBM Spectrum Scale: Problem Determination Guide*.

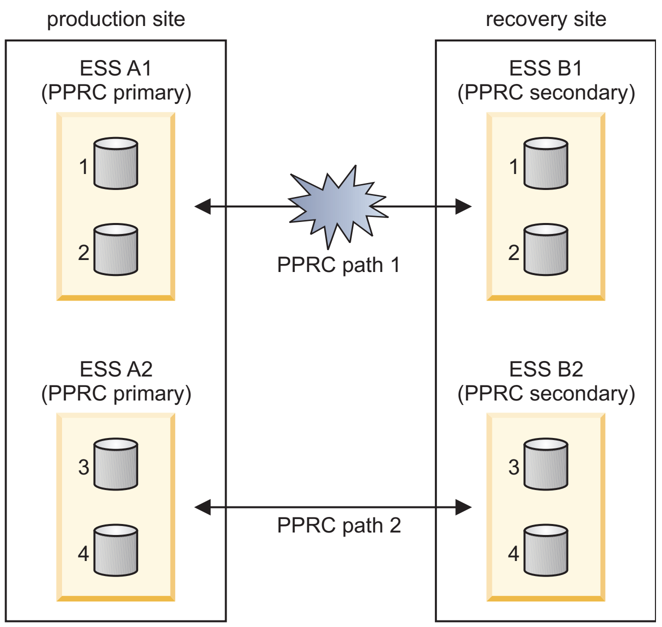


Figure 8. Violation of write ordering without the use of a PPRC consistency group

Asynchronous mirroring with ESS FlashCopy

The FlashCopy feature of ESS provides an easy implementation to make a point-in-time copy of a GPFS file system as an online backup mechanism. This function provides an instantaneous copy of the original data on the target disk, while the actual transfer of data takes place asynchronously and is fully transparent to the user.

When a FlashCopy disk is first created, the subsystem establishes a control bitmap that is subsequently used to track the changes between the source and the target disks. When processing read I/O requests sent to the target disk, this bitmap is consulted to determine whether the request can be satisfied using the target's copy of the requested block. If the track containing the requested data has not yet been copied, the source disk is instead accessed and its copy of the data is used to satisfy the request. The FlashCopy feature is further described in the *IBM Enterprise Storage Server*. Similar to PPRC, FlashCopy operations can be invoked using the ESS Copy Services web interface or the command-line scripts.

To prevent the appearance of out-of-order updates, it is important to consider data consistency when using FlashCopy. Prior to taking the FlashCopy image all disk volumes that make up the file system must be brought to same logical point in time. Two methods may be used to provide for data consistency in the FlashCopy image of your GPFS file system. Both techniques guarantee the consistency of the FlashCopy image by the means of temporary suspension of I/O, but either can be seen as the preferred method depending on your specific requirements and the nature of your GPFS client application:

Several uses of the FlashCopy replica after its initial creation can be considered. For example, if your primary operating environment suffers a permanent loss or a corruption of data, you may choose to flash the target disks back onto the originals to quickly restore access to a copy of the file system as seen at the time of the previous snapshot. Before restoring the file system from a FlashCopy, please make sure to suspend the activity of the GPFS client processes and unmount the file system on all GPFS nodes.

To protect your data against site-wide disasters, you may chose to instead transfer the replica offsite to a remote location using PPRC or any other type of bulk data transfer technology. Alternatively, you may choose to mount the FlashCopy image as a separate file system on your backup processing server and transfer the data to tape storage. To enable regular file system access to your FlashCopy replica, create a single-node GPFS cluster on your backup server node and execute the **mmfsctl syncFSconfig** command to import the definition of the file system from your production GPFS cluster. Figure 9 provides a schematic view of an asynchronous recovery GPFS environment using a combination of PPRC and FlashCopy. In such environments, two independent GPFS clusters are set up in distinct geographic locations (production and recovery sites). We refer to such clusters as *peer* GPFS clusters. Peer clusters must share the same UID/GID space, but otherwise need not belong to the same administrative domain. In particular, the administrator is free to identify nodes in both clusters with the same set of short hostnames if such a configuration is indeed desired.

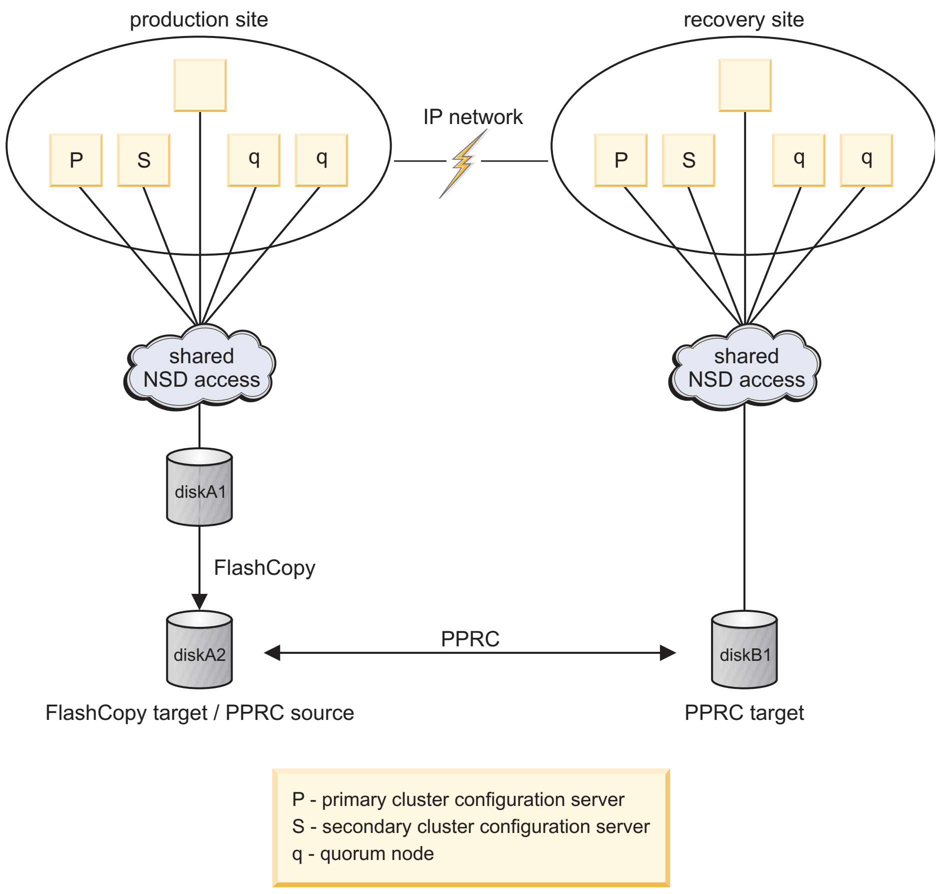


Figure 9. High-level organization of a FlashCopy/PPRC recovery environment

FlashCopy/PPRC provides for the availability of the file system's on-disk content in the recovery cluster. But in order to make the file system known and accessible, you must issue the **mmfsctl syncFSConfig** command to:

- Import the state of the file system's configuration from the primary location.
- Propagate all relevant changes to the configuration in the primary cluster to its peer to prevent the risks of discrepancy between the peer's **mmsdrfs** file and the content of the file system descriptor found in the snapshot.

  It is suggested you generate a new FlashCopy replica immediately after every administrative change to the state of the file system. This eliminates the risk of a discrepancy between the GPFS configuration data contained in the **mmsdrfs** file and the on-disk content of the replica.

Restriction: The primary copy of a GPFS file system and its FlashCopy image cannot coexist in the same GPFS cluster. A node can mount either the original copy of the file system or one of its FlashCopy replicas, but not both. This restriction has to do with the current implementation of the NSD-to-LUN mapping mechanism, which scans all locally-attached disks, searching for a specific value (the NSD id) at a particular location on disk. If both the original volume and its FlashCopy image are visible to a particular node, these disks would appear to GPFS as distinct devices with identical NSD ids

For this reason, we ask users to zone their SAN configurations such that at most one replica of any given GPFS disk is visible from any node. That is, the nodes in your production cluster should have access to the disks that make up the actual file system but should not see the disks holding the FlashCopy images, whereas the backup server should see the FlashCopy targets but not the originals.

Alternatively, you can use the **nsddevices** user exit located in **/var/mmfs/etc/** to explicitly define the subset of the locally visible disks to be accessed during the NSD device scan on the local node. See "Setting up FlashCopy using file-system-level suspension."

In the production GPFS cluster, FlashCopy is used to take periodic volume-level snapshots of the GPFS file system onto a set of idle local disks and the snapshots are then propagated to the peer recovery cluster using PPRC. The goal of this technique is to provide a consistent (but not necessarily up-to-date) image of the file system at the recovery site that can be used to restore operation in the event of a disaster in the primary cluster. Note that since from the GPFS perspective, the replicas are two entirely distinct file systems, nothing prevents the administrator from mounting and operating on both replicas concurrently if deemed necessary.

Setting up FlashCopy using file-system-level suspension

The following example illustrates using file-system-level suspension to provide for data consistency.

To prepare a file system as depicted in Figure 9 on page 104, using file-system-level suspension to provide for data consistency, consider the configuration:

Site A - primary cluster
Consisting of:
- Nodes – **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**, **nodeA005**
- Disk device names – **diskA1**, **diskA2**

Site B - recovery site
Consisting of:
- Nodes – **nodeB001**, **nodeB002**, **nodeB003**, **nodeB004**, **nodeB005**
- Disks – **diskB1**

There is a single file system, **fs0**, defined on **diskA1**. To create a volume-level snapshot and propagate the snapshot to the recovery cluster:

1. Define an **nsddevices** user exit file to prevent the production site from using the FlashCopy target disk **diskA2**:
   ```
   echo "echo diskA1 hdisk" > /var/mmfs/etc/nsddevices
   chmod 744 /var/mmfs/etc/nsddevices
   ```
   Refer to the prolog of **/usr/lpp/mmfs/samples/nsddevices.samples** for detailed instructions on the usage of **nsddevices**.
2. In the primary cluster, suspend all file system I/O activity and flush the GPFS buffers:
   ```
   mmfsctl fs0 suspend
   ```
3. Establish a FlashCopy pair using **diskA1** as the source and **diskA2** as the target. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
4. Resume the file system I/O activity:
   ```
   mmfsctl fs0 resume
   ```
5. Establish a PPRC path and a synchronous PPRC volume pair **diskA2-diskB** (primary-secondary). Use the **copy entire volume** option and leave the **permit read from secondary** option disabled. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.

6. Wait for the completion of the FlashCopy background task. Wait for the PPRC pair to reach the *duplex* (fully synchronized) state.
7. Terminate the PPRC volume pair **diskA2-diskB**. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
8. If this is the first time the snapshot is taken, or if the configuration state of **fs0** changed since the previous snapshot, propagate the most recent configuration to site **B**:

   ```
   mmfsctl all syncFSconfig -n nodes.siteB
   ```

   The file **nodes.siteB** lists all of the nodes, one per line, at the recovery site.

Chapter 7. Implementing a clustered NFS environment on Linux

In addition to the traditional exporting of GPFS file systems using the Network File System (NFS) protocol, GPFS allows you to configure a subset of the nodes in the cluster to provide a highly-available solution for exporting GPFS file systems using NFS.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

The participating nodes are designated as Cluster NFS (CNFS) member nodes and the entire setup is frequently referred to as CNFS or a CNFS cluster.

In this solution, all CNFS nodes export the same file systems to the NFS clients. When one of the CNFS nodes fails, the NFS serving load moves from the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover. For the NFS client node to experience a seamless failover, hard mounts must be used. The use of soft mounts will likely result in stale NFS file handle conditions when a server experiences a problem, even though CNFS failover will still be done.

Currently, CNFS is supported only in the Linux environment. For an up-to-date list of supported operating systems, specific distributions, and other dependencies, refer to the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

NFS monitoring

Every node in the CNFS cluster runs a separate GPFS utility that monitors GPFS, NFS, and networking components on the node. Upon failure detection and based on your configuration, the monitoring utility might invoke a failover.

While an NFS server is in a grace period, the NFS monitor sets the server's NFS state to "Degraded". For information about CES states, see the topic *CES monitoring and troubleshooting* in the *IBM Spectrum Scale: Problem Determination Guide*.

NFS failover

As part of GPFS recovery, the CNFS cluster failover mechanism is invoked. It transfers the NFS serving load that was served by the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover.

The failover mechanism is based on IP address failover. The CNFS IP address is moved from the failing node to a healthy node in the CNFS cluster. In addition, it guarantees NFS lock (NLM) recovery.

Failover processing may involve rebooting of the problem node. To minimize the effects of the reboot, it is recommended that the CNFS nodes be dedicated to that purpose and are not used to run other critical processes. CNFS node rebooting should not be disabled or the failover reliability will be severely impacted.

NFS locking and load balancing

CNFS supports a failover of all of the node’s load together (all of its NFS IP addresses) as one unit to another node. However, if no locks are outstanding, individual IP addresses can be moved to other nodes for load balancing purposes.

CNFS is dependent on DNS for any automated load balancing of NFS clients among the NFS cluster nodes. Using the round-robin algorithm is highly recommended.

CNFS network setup

In addition to one set of IP addresses for the GPFS cluster, a separate set of one or more IP addresses is required for CNFS serving.

The GPFS cluster can be defined over an IPv4 or IPv6 network. The IP addresses specified for CNFS can also be IPv4 or IPv6. The GPFS cluster and CNFS are not required to be on the same version of IP, but IPv6 must be enabled on GPFS to support IPv6 on CNFS.

CNFS setup

You can set up a clustered NFS environment within a GPFS cluster.

To do this, follow these steps:

1. Designate a separate directory for the CNFS shared files:
   ```
   mmchconfig cnfsSharedRoot=directory
   ```
   where:

   cnfsSharedRoot=*directory*
   : Is the path name to a GPFS directory, preferably on a small separate file system that is not exported by NFS. The GPFS file system that contains the directory must be configured to be mounted automatically upon GPFS start on each of the CNFS nodes (**-A yes** option on the **mmchfs** command). **cnfsSharedRoot** is a mandatory parameter and must be defined first.
2. Add all GPFS file systems that need to be exported to **/etc/exports**. For NSF export considerations, see the topic *Exporting a GPFS file system using NFS* in the *IBM Spectrum Scale: Administration and Programming Reference*.
3. If the shared directory from step 1 is in an exported file system, restrict access to that directory.
4. Use the **mmchnode** command to add nodes to the CNFS cluster:
   ```
   mmchnode --cnfs-interface=ip_address_list -N node
   ```
   where:

   ip_address_list
   : Is a comma-separated list of host names or IP addresses to be used for GPFS cluster NFS serving.

   node
   : Identifies a GPFS node to be added to the CNFS cluster.

   For more information, see the topic *mmchnode command* in the *IBM Spectrum Scale: Administration and Programming Reference*.
5. Use the **mmchconfig** command to configure the optional CNFS parameters.

   cnfsMountdPort=*mountd_port*
   : Specifies the port number to be used for the **rpc.mountd** daemon.

     For CNFS to work correctly with the automounter (AMD), the **rpc.mountd** daemon on the different nodes must be bound to the same port.

   cnfsNFSDprocs=*nfsd_procs*
   : Specifies the number of **nfsd** kernel threads. The default is 32.

cnfsVersions=*nfs_versions*
Specifies a comma-separated list of protocol versions that CNFS should start and monitor. The default is **3,4**. If you are not using NFS v3 and NFS v4, specify this parameter with the appropriate values for your configuration.

Note: If you are not using NFS v3 and NFS v4, and you do not explicitly specify **cnfsVersions** with the protocol versions on your system, the following message will continually appear in the **mmfs.log**:

```
Found NFS version mismatch between CNFS and current running config, check the OS config files.
```

6. If multiple failover groups are desired, assign a group ID to each NFS node:

   mmchnode --cnfs-groupid=*groupid* **-N** *node*

   To assign NFS nodes to different groups, use a group ID that is in a different range of ten. For example, a node with group ID 2*n* will fail over only to nodes in the same range of ten (which means any node with group ID 20 to 29). Failover in the same group will first look for one of the nodes with the same group ID. If none are found, any node in the group range starting at *n*0 to *n*9 is selected.

CNFS administration

There are some common CNFS administration tasks in this topic along with a sample configuration.

To query the current CNFS configuration, enter:

mmlscluster --cnfs

To temporarily disable CNFS on one or more nodes, enter:

mmchnode --cnfs-disable -N *NodeList*

Note: This operation affects only the high-availability aspects of the CNFS functionality. Normal NFS exporting of the data from the node is not affected. All currently defined CNFS IP addresses remain unchanged. There will be no automatic failover from or to this node in case of a failure. If failover is desired, GPFS should be shut down on the affected node prior to issuing the **mmchnode** command.

To re-enable previously-disabled CNFS member nodes, enter:

mmchnode --cnfs-enable -N *NodeList*

Note: If the GPFS daemon is running on a node on which CNFS is being re-enabled, the node will try to activate its CNFS IP address. If the IP address was currently on some other CNFS-enabled node, that activation would include a takeover.

To permanently remove nodes from the CNFS cluster, enter:

mmchnode --cnfs-interface=DELETE -N *NodeList*

Note: This operation affects only the high-availability aspects of the CNFS functionality. Normal NFS exporting of the data from the node is not affected. All currently defined CNFS IP addresses remain unchanged. There will be no automatic failover from or to this node in case of a failure. If failover is desired, GPFS should be shut down on the affected node prior to issuing the **mmchnode** command.

A sample CNFS configuration

Here is a CNFS configuration example, which assumes the following:

- Your GPFS cluster contains three nodes: `fin18`, `fin19`, and `fin20`
- The host names for NFS serving are: `fin18nfs`, `fin19nfs`, and `fin20nfs`

To define a CNFS cluster made up of these nodes, follow these steps:

1. Add the desired GPFS file systems to **/etc/exports** on each of the nodes.

2. Create a directory called **ha** in one of the GPFS file systems by entering:

```
mkdir /gpfs/fs1/ha
```

3. Create a temporary file called **/tmp/hanfs-list**, which contains the following lines:

```
fin18 --cnfs-interface=fin18nfs
fin19 --cnfs-interface=fin19nfs
fin20 --cnfs-interface=fin20nfs
```

4. Set the CNFS shared directory by entering:

```
mmchconfig cnfsSharedRoot=/gpfs/fs1/ha
```

5. Create the CNFS cluster with the **mmchnode** command, by entering:

```
mmchnode -S /tmp/hanfs-list
```

6. Access the exported GPFS file systems over NFS. If one or more GPFS nodes fail, the NFS clients should continue uninterrupted.

Chapter 8. Implementing Cluster Export Services

Cluster Export Services (CES) provides highly available file and object services to a GPFS cluster by using Network File System (NFS), Object, or Server Message Block (SMB) protocols.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

CES is an alternate approach to using a clustered Network File System (NFS) to export GPFS file systems. For more information about CES and protocol configuration, see *IBM Spectrum Scale: Administration and Programming Reference*.

CES features

To successfully use Cluster Export Services (CES), you must consider function prerequisites, setup and configuration, failover/failback policies, and other management and administration requirements.

CES cluster setup

You can set up a Cluster Export Services (CES) environment within a GPFS cluster.

The CES shared root (cesSharedRoot) directory is needed for storing CES shared configuration data, protocol recovery, and some other protocol-specific purposes. It is part of the Cluster Export Configuration and is shared between the protocols. Every CES node requires access to the path that is configured as shared root.

To update the CES shared root directory, you must shut down the cluster, set the CES shared root directory, and start the cluster again:

```
mmshutdown -a
mmchconfig cesSharedRoot=shared_root_path
mmstartup -a
```

The recommendation for CES shared root directory is a dedicated file system. It can also reside in an existing GPFS file system. In any case, the CES shared root directory must be on GPFS and must be available when it is configured through the **mmchconfig** command.

To enable protocol nodes, the CES shared root directory must be defined. To enable protocol nodes, use the following command:

```
mmchnode --ces-enable -N Node1[,Node2...]
```

To disable a CES node, use the following command:

```
mmchnode --ces-disable -N Node1[,Node2...]
```

Preparing to perform service actions on the CES shared root directory file system

The CES shared root directory file system must be kept available for protocols operation to function. If a service action is to be performed on the CES shared root directory file system, perform the steps that follow.

Commands such as **mmshutdown, mmstartup and mmmount**, can be passed in the `cesnodes` node class parameter to ensure operation on all protocol nodes.

The following steps are used to perform service actions on the CES shared root filesystem :

1. Inform users of the impact to protocols. Quiesce protocol related I/O and mounts if possible. Quiesce cluster functions in progress on protocol nodes such as recalls, migrations, AFM, backup, and any policies that may be in use on the protocol nodes; or transition these cluster functions to other nodes. Finally, verify that file system quorum can be achieved by the remaining cluster nodes.
2. Shut down GPFS on all protocol nodes:
   ```
   mmshutdown -N cesnodes
   ```
   Note: Only protocol nodes need to be shut down for service of the CES shared root directory file system. However, other nodes may need to unmount the file system, depending on what service is being performed.

Protocol nodes are now ready for service actions to be performed on the CES shared root directory or the nodes themselves. To recover from a service action:
1. Start up GPFS on all protocol nodes:
   ```
   mmstartup -N cesnodes
   ```
2. Make sure that the CES shared root directory file system is mounted on all protocol nodes:
   ```
   mmmount cesSharedRoot -N cesnodes
   ```
3. Verify that all protocol services have been started:
   ```
   mmces service list -a
   ```

CES network configuration

Cluster Export Services (CES) IP addresses are used to export data via the NFS, SMB, and Object protocols. File and Object clients use the public IPs to access data on GPFS file systems.

CES IP addresses have the following characteristics:
- Shared between all CES protocols
- Organized in an *address pool* (there can be fewer or more CES addresses than nodes)
- Hosted on the CES nodes (there can be CES nodes without CES addresses)
- Can move between CES nodes (triggered manually via the command or as part of a CES failover)
- Must not be used for GPFS communication at the same time

CES IP addresses have these restrictions:
- The network on CES nodes must be configured so that all CES IPs can run on any CES node. Typically this configuration requires that all CES nodes have at least one NIC interface or VLAN-compatible interface with each CES IP network address.
- CES IPs are created as aliases on each CES node. Do not include the primary address of an adapter in the CES IP address pool.
- CES IPs must be resolvable by DNS or `/etc/hosts`.
- CES does not manage the subnet or netmask configuration; it is the user's task.

To add CES IP addresses to the address pool, use the **mmces** command:

```
mmces address add --ces-ip Address[,Address...]
```

By default, addresses are distributed among the CES nodes, but a new address can be assigned to a particular node:

```
mmces address add --ces-ip Address[,Address...] --ces-node Node
```

After a CES IP address is added to the address pool, you can manually move the address to a particular node:

```
mmces address move --ces-ip Address[,Address...] --ces-node Node
```

You can remove a CES IP address from the address pool with the **mmces** command:

```
mmces address remove --ces-ip Address[,Address...]
```

Removing an address while there are clients connected causes the clients to lose those connections. Any reconnection to the removed IP results in a failure. If DNS is used to map a name entry to one or more IP addresses, update the DNS to ensure that a client is not presented an address that was already removed from the pool. This process might also include invalidation of any DNS caches.

CES addresses are virtual IP addresses

The CES addresses that are assigned to the CES nodes are implemented as IP aliases. Each network adapter that hosts CES addresses must already be configured (with different non-CES IPs) in `/etc/sysconfig`. CES uses the netmask to figure out which interfaces to use. For example, if `eth1` is 10.1.1.1 and `eth2` is 9.1.1.1, then the CES IP 10.1.1.100 maps to `eth1` and the CES IP 9.1.1.100 maps to `eth2`.

Virtual IP addresses include the following advantages:

- The node does not need to wait for the switch to accept the link when an IP address is failed back. Since the address is an alias, the interface on which it resides is already up.
- IP address failover is much faster.
- Administration is simplified by providing a clear distinction between the *system IP* and the *CES IP*. For example, you have a two-node cluster. One of the nodes in the two-node cluster has a problem that induces failover and someone logs in to the suspected node to reboot it. The surviving node might get rebooted by accident if the system address was migrated to the surviving node.

How to use an alias

To use an alias address for CES, you need to provide a static IP address that is not already defined as an alias in the `/etc/sysconfig/network-scripts` directory.

Before you enable the node as a CES node, configure the network adapters for each subnet that are represented in the CES address pool:

1. Define a static IP address for the device:

   ```
   /etc/sysconfig/network-scripts/ifcfg-eth0
   DEVICE=eth1
   BOOTPROTO=none
   IPADDR=10.1.1.10
   NETMASK=255.255.255.0
   ONBOOT=yes
   GATEWAY=10.1.1.1
   TYPE=Ethernet
   ```

2. Ensure that there are no aliases that are defined in the network-scripts directory for this interface:

   ```
   # ls -l /etc/sysconfig/network-scripts/ifcfg-eth1:*
   ls: /etc/sysconfig/network-scripts/ifcfg-eth1:*: No such file or directory
   ```

After the node is enabled as a CES node, no further action is required. CES addresses are added as aliases to the already configured adapters.

CES address failover and distribution policies

When a Cluster Export Services (CES) node leaves the GPFS cluster, any CES IP addresses that are assigned to that node are moved to CES nodes still within the cluster. Additionally, certain error conditions and administrative operations can cause a node to release its addresses to be reassigned to other nodes.

As CES nodes enter and leave the GPFS cluster, the addresses are distributed among the nodes according to the address distribution policy that is selected. In addition, you can disable automatic address distribution to allow the user to manually maintain the address-to-node assignments.

The address distribution policy is set with the **mmces** command:

```
mmces address policy [even-coverage | balanced-load | node-affinity | none]
```

The following list describes each type of address distribution policy:

even-coverage
: Distributes the addresses among the available nodes. The even-coverage policy is the default address distribution policy.

balanced-load
: Distributes the addresses to approach an optimized load distribution. The load (network and CPU) on all the nodes are monitored. Addresses are moved based on given policies for optimized load throughout the cluster.

node-affinity
: Attempts to keep an address on the node to which the user manually assigned it. If the **mmces address add** command is used with the `--node` option, the address is marked as being associated with that node. Similarly, if an address is moved with the **mmces address move** command, the address is marked as being associated with the destination node. Any automatic movement, such as reassigning a down node's addresses, does not change this association. Addresses that are enabled with no node specification do not have a node association.

  Addresses that are associated with a node but assigned to a different node are moved back to the associated node if possible.

Automatic address distribution is performed in the background in a way as to not disrupt the protocol servers more than necessary. If you want immediate redistribution of the addresses, use the **mmces** command to force an immediate rebalance:

```
mmces address move --rebalance
```

You can further control the assignment of CES addresses by placing nodes or addresses in CES groups. For more information, see the topic *Configuring CES protocol service IP addresses* in the *IBM Spectrum Scale: Administration and Programming Reference*.

CES protocol management

Cluster Export Services (CES) protocols (NFS, SMB, and Object) are enabled or disabled with the **mmces** command.

Command examples:

```
mmces service enable [NFS | OBJ | SMB]
mmces service disable [NFS | OBJ | SMB]
```

After a protocol is enabled, the protocol is started on all CES nodes.

When a protocol is disabled, the protocol is stopped on all CES nodes and all protocol-specific configuration data is removed.

CES management and administration

Cluster Export Services (CES) nodes can be suspended for maintenance reasons with the **mmces** command.

For example:

```
mmces node suspend [-N Node[,Node...]
```

When a node is suspended:

- The GPFS state of the node is unaffected. It remains an active member of the cluster.
- All CES IP addresses that are assigned to the node are reassigned to other nodes. Assignment of addresses to a suspend node is not allowed.
- All CES monitoring operations are stopped.
- Servers for the enabled CES protocols continue to run, but can be stopped.
- Suspended protocol servers can be stopped and started on the node with the following **mmces** commands.

  ```
  mmces service stop [NFS | OBJ | SMB] [-N Node[,Node...]]
  mmces service start [NFS | OBJ | SMB] [-N Node[,Node...]]
  ```
- A node or a group of nodes can be suspended and resume normal operation with the following **mmces** commands.

  ```
  mmces node suspend -N node1,node2,node3
  mmces node resume
  ```

  After a node is resumed, monitoring on the node is started and the node is eligible for address assignments.

CES NFS support

In Cluster Export Services (CES), you must consider monitoring, service and export configuration, failover policies, migration, and client support requirements for Network File System (NFS) support.

NFS support levels

NFS versions 3 (NFSv3) and 4 (NFSv4.0) are supported.

NFS monitoring

The NFS servers are monitored to check for proper functions. If a problem is found, the CES addresses of the node are reassigned, and the node state is set to `failed`. When the problem is corrected, the node resumes normal operation.

NFS service configuration

Configuration options for the NFS service can be set with the **`mmnfs configuration`** command.

You can use the **`mmnfs configuration`** command to set and list default settings for NFS such as the port number for the NFS service, the default access mode for exported file systems, the log level, and enable or disable status for delegations. For a list of configurable attributes, see the topic *mmnfs command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Some of the attributes such as the protocol can be overruled for a given export on a per-client base. For example, the default settings might have NFS protocols 3 and 4 enabled, but the export for a client might restrict it to NFS version 4 only.

NFS export configuration

Exports can be added, removed, or changed with the **`mmnfs`** export command. Authentication must be set up before you define an export.

Exports can be declared for any directory in the GPFS file system, including a fileset junction. At the time where exports are declared, these folders must exist physically in GPFS. Only folders in the GPFS file system can be exported. No folders that are located only locally on a server node can be exported because they cannot be used in a failover situation.

Export-add and export-remove operations can be applied at run time of the NFS service. The export-change operation does require a restart of the NFS service on all server nodes that is followed by a 60-second grace period to allow connected clients to reclaim their locks and to avoid concurrent lock requests from new clients.

NFS failover

When a CES node leaves the cluster, the CES addresses assigned to that node are redistributed among the remaining nodes. Remote clients that access the GPFS file system might see a pause in service while the internal state information is passed to the new servers.

NFS clients

When you work with NFS clients, consider the following points:

- If you mount the same NFS export on one client from two different IBM Spectrum Scale NFS protocol nodes, data corruption might occur.
- Cross-protocol notifications are not supported by clustered NFS. For example, files that are created with NFS are not automatically visible on SMB clients and SMB clients need to refresh the directory listing by performing a manual refresh in Microsoft Windows Explorer. Similarly, files that are created from other methods such as files that are created with FTP or files that are restored from a backup are not automatically visible.
- The NFS protocol version that is used as the default on a client operating system might differ from what you expect. If you are using a client that mounts NFSv3 by default, and you want to mount NFSv4, then you must explicitly specify NFSv4 in the **`mount`** command. For more information, see the **`mount`** command for your client operating system.
- A client must mount an NFS export by using the IP address of the GPFS system. If a host name is used, ensure that the name is unique and remains unique.

  If a DNS Round Robin (RR) entry name is used to mount an NFSv3 export, data corruption and data unavailability might occur. The lock manager on the GPFS file system is not *clustered-system-aware*.
- Reusing an export ID might cause unexpected behavior on a running system. When an export is removed, avoid reusing the export ID either for a different export or for reexporting the same file system.

Coexistence with clustered NFS and migration

CES and clustered NFS cannot exist in the same GPFS cluster.

Choosing between CNFS or CES

If you want to put highly available NFS services on top of the GPFS file system, you have the choice between clustered NFS (Chapter 7, “Implementing a clustered NFS environment on Linux,” on page 107) and Cluster Export Services (Chapter 8, “Implementing Cluster Export Services,” on page 111).

To help you choose one of these NFS offerings, consider the following points:

Multiprotocol support

If you plan to use other protocols (such as SMB or Object) in addition to NFS, CES must be chosen. While CNFS provides support only for NFS, the CES infrastructure adds support also for SMB and Object. With CES, you can start with NFS and add (or remove) other protocols at any time.

Command support

While CNFS provides native GPFS command support for creation and management of the CNFS cluster, it lacks commands to manage the NFS service and NFS exports. The CES infrastructure introduces native GPFS commands to manage the CES cluster. Furthermore, there are also commands to manage the supported protocol services and the NFS exports. For example, with CES, you do not need to adapt NFS configuration files individually on the protocol nodes. This work is done by the new GPFS commands that are provided for CES.

Performance

CNFS is based on the kernel NFS server while NFS support in CES is based on the Ganesha NFS server operating in user space. Due to the different nature of these NFS I/O stacks, performance depends on system characteristics and NFS workload. Contact your IBM representative to get help with sizing the required number of protocol nodes to support certain workload characteristics and protocol connection limits.

There is no general answer about which of the two NFS servers performs better as this depends on many factors. Tests that are conducted with both NFS I/O stacks over various workloads show that the kernel-based NFS server (CNFS) performs better under metadata-intensive workloads. Typically this testing is with many smaller files and structures. The Ganesha NFS server provides better performance on other data-intensive workloads such as video streaming.

Note: CES provides a different interface to obtain performance metrics for NFS. CNFS uses the existing interfaces to obtain NFS metrics from the kernel (such as `nfsstat` or the `/proc` interface). The CES framework provides the **`mmperfmon query`** command for Ganesha-based NFS statistics. For more information, see the topic *mmperfmon command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Migration of CNFS to CES

For information about migrating existing CNFS environments to CES, see “Migration of CNFS clusters to CES clusters” on page 119.

CES SMB support

In GPFS 4.1.1 and later, you can access a GPFS file system with an SMB client using its inherent SMB semantics.

The following features are provided:

Clustered SMB support

SMB clients can connect to any of the protocol nodes and get access to the shares defined. A clustered registry makes sure that all nodes see the same configuration data. Therefore, clients can connect to any Cluster Export Services (CES) node and see the same data. Moreover, the state of opened files (share modes, open modes, access masks, locks, and so on) is also shared among the CES nodes so that data integrity is maintained. On failures, clients can reconnect transparently to another cluster node as the IP addresses of failing nodes are transferred to another cluster node.

The supported protocol levels are SMB2 and the base functions of SMB3 (dialect negotiation, secure negotiation, encryption of data on the wire).

Export management command

With the **`mmsmb`** command, IBM Spectrum Scale provides a comprehensive entry point to manage all SMB-related configuration tasks like creating, changing, and deleting SMB shares.

SMB monitoring

The monitoring framework detects issue with the SMB services and triggers failover in case of an unrecoverable error.

Integrated installation

The SMB services are installed by the integrated installer together with the CES framework and the other protocols NFS and Object.

SMB performance metrics

The SMB services provide two sets of performance metrics that are collected by the performance monitor framework. Both current and historic data (with lower granularity) can be retrieved. The two sets of metrics are global SMB metrics (like number of connects and disconnects) and metrics on for each SMB request (number, time, throughput). The mmperfmon query tool provides access to the most important SMB metrics via predefined queries. Moreover, metrics for the clustered file metadata database CTDB are collected and exposed via the **mmperfmon query** command.

Cross-protocol integration with NFS

In IBM Spectrum Scale 4.1.1 and later, you can concurrently access the same file data with SMB, NFS, and native POSIX access. You cannot concurrently access the same file data using traditional file protocols.

Authentication and mapping

The SMB services can be configured to authenticate against the authentication services MS Active Directory and LDAP. Mapping MS security identifiers (SIDs) to the POSIX user and group IDs on the file server can either be done automatically by using the so-called autorid mechanism or external mapping services like RFC 2307 or MS Services for Unix. If none of the offered authentication and mapping schemes matches the environmental requirements, a user-defined configuration can be established.

CES OBJ support

In Cluster Export Services (CES), you must consider several types of requirements for Object (OBJ) support.

Openstack support levels

Swift (Kilo version) and Keystone (Keystone V3) are supported.

Object monitoring

The Object servers are monitored for proper functioning. If a problem is found, the CES addresses of the node are reassigned, and the node state is set to failed. When the problem is corrected, the node resumes normal operation.

Object service configuration

The Object service configuration is controlled by the respective Swift and Keystone configuration files. The main versions of these files are stored in the CCR repository, and copies exist in the `/etc/swift` and `/etc/keystone` directories. The files that are stored in those directories should not be directly modified since they are overwritten by the files that are stored in the CCR. To change the Swift or Keystone configuration, use the **mmobj config change** command to modify the configuration files that are stored in the CCR. The monitoring framework is notified of the change and propagates the file to the local file system of the CES nodes. For information about the values that can be changed and their associated function, refer to the administration guides for Swift and Keystone.

To change the authentication that is used by the Keystone server, use the **mmuserauth** command to change the authentication repository to AD or LDAP, or to enable SSL communication to the Keystone server.

Object fileset configuration

A separate fileset must be created for the Object service. This fileset is automatically created in the GPFS file system that is specified during installation. Evaluate the data that is expected to be stored by the Object service to determine the required number of inodes that are needed, which can also be specified during installation.

Object failover

When a CES node leaves the cluster, the CES addresses that are assigned to that node are redistributed among the remaining nodes. Remote clients that access the Object service might see active connections drop or a pause in service while the internal state information is passed to the new servers. Clients with active connections to the CES addresses that are migrated might have their connections unexpectedly drop. Clients are expected to retry their requests when this happens.

Certain Object-related services can be migrated when a node is taken offline. If the node was hosting the backend database for Keystone or certain Swift services that are designated as singletons (such as the auditor), those services are started on the active node that received the associated CES addresses of the failed node. Normal operation of the Object service resumes after the CES addresses are reassigned and necessary services automatically restarted.

Object clients

The Object service is based on Swift and Keystone, and externalizes their associated interfaces. Clients should follow the associated specifications for those interfaces. Clients should expect dropped connections or delays during CES node failover. In such situations, clients should retry the request or allow more time for the request to complete.

To connect to an Object service, clients should use a DNS service that returns a single CES address from a pool of addresses or connect to a load balancer that distributes requests among the CES nodes. Clients should not use hard-coded CES addresses to connect to Object services. For example, the authentication URL should refer to a DNS host name such as `http://protocols.gpfs.net:35357/v3` rather than a CES address.

Migration of CNFS clusters to CES clusters

If your system has established clustered Network File System (NFS) clusters, you might consider migrating these clusters to Cluster Export Services (CES) clusters.

Points to consider before you migrate

Cluster Export Services (CES) protocol nodes have the following dependencies and restrictions:

- CES nodes cannot coexist with CNFS clusters.
- CES NFS does not support the concept of failover groups. If this feature in CNFS is important to you, you might want to reconsider migrating to CES NFS until you no longer need this feature.
- Spectrumscale installer confirms that your cluster does not have an existing CNFS cluster.
- CES nodes use SMB, Ganesha NFS, and Openstack SWIFT Object services.
- File system ACL permissions need to be in NFSv4 format.

- CES NFS (Ganesha) services expects NFSv4 ACL formats.
- CES SMB (Samba) services expects NFSv4 ACL formats
- Existing CNFS exports definitions are not compatible with Ganesha NFS. It is best to script and automate the creation of the equivalent exports by using the **mmnfs export create** command to reduce the amount you need to change in the future.
- CES nodes need authentication that is configured.
- CES nodes do not support IP failover groups (as found in CNFS).
- There is a maximum of 16 protocol nodes in a CES cluster if the SMB protocol is also enabled.
- There is a maximum of 32 protocol nodes in a CES cluster if only NFS is enabled.

Because there is a mutual exclusivity between CNFS and CES nodes, you need to accommodate user and application access outage while CES clusters nodes are installed, configured, set up for authentication, and the NFS exports are re-created. The duration of this process depends on the complexity of the customer environment.

You might want to procure new CES nodes or reuse the existing CNFS nodes. Either way, you cannot use the **spectrumscale** installation toolkit until the CNFS nodes are unconfigured.

If you do not have an opportunity to test or plan the implementation of a CES cluster elsewhere, you might have to deal with the design and implementation considerations and issues during the planned outage period. Usually this process is straightforward and quick. If you have a more complex environment, however, it might take longer than the allotted upgrade window to complete the migration. In this case, it is possible to set up one or two non-CNFS, NFS servers to serve NFS for a short time. During this time, you would move all your CNFS IPs to these nodes as you decommission the CNFS cluster. Then, after you successfully set up your CES nodes, authentication, and corresponding exports, you can move the IPs from the temporary NFS servers over to the CES nodes.

Saving CNFS export configuration

You need to make a copy of the exports configuration file `/etc/exports` so that you can use this file as the basis for creating the new exports in Ganesha NFS. Ganesha NFS exports configuration needs to be created by using the **mmnfs export create** command or created in bulk by using the **mmnfs export load** command. When you unconfigure CNFS, you also need to delete the `/etc/exports` file from each of the CNFS nodes.

Steps to unconfigure CNFS

1. CES nodes are configured to run only on RHEL 7.x nodes. If you are planning to convert your existing CNFS nodes to CES nodes, ensure that they are upgraded to RHEL 7.x first. It is best to upgrade the nodes first while they are running CNFS so that you can ensure that functions are the same as before you start to change over to CES nodes.
2. Ensure that you stop application and user access to the CNFS exports
3. Run the **mmchnode** command to dereference / evict a CNFS node from the cluster. This command removes both the node and its associated IP):
   ```
   mmchnode –cnfs-interface=default –N node1Name,node2Name,...
   ```
4. When you remove the last node, CNFS is unconfigured and you see an output similar to this result:
   ```
   [root@esnode3 ~]# mmlscluster --cnfs

   GPFS cluster information
   ========================
     GPFS cluster name:        esvcluster1.esnode1
     GPFS cluster id:          15635445795275488305

   mmlscluster:  CNFS is not defined in this cluster.

   [root@esnode3 ~]#
   ```

5. Consider de-refencing the GPFS variable *cnfsSharedRoot*, although this step is not a requirement.
6. You can now delete the `/etc/exports` file on each of the CNFS nodes. Ensure that you have a backup copy of this file to use as a reference when you create the exports under CES NFS.

Steps to Configure CES NFS

1. If you have not yet configured the CES nodes for authentication, complete this step before you create the exports. Refer to "CES NFS support" on page 115 for details on configuring authentication for your environment.
2. Ensure that the file systems you want to export access to are configured for the NFSv4 security model. If you are converting an existing file system from another security model to NFSv4 you might need to review the ACL structures of the files and verify that your access will work as expected.
3. If you have special configurations or options set in CNFS server, you might also want to reflect these settings in Ganesha NFS. You need to review the appropriateness of these settings for the new environment. To change the settings, use the following command:
   ```
   mmnfs configuration change
   ```
4. If you have many exports to be converted to CES NFS, use the following command:
   ```
   mmnfs  export load exportCfgFile
   ```
   exportCfgFile contains a listing of all your exports as defined in the format that is used for `/etc/ganesha/gpfs.ganesha.exports.conf`.
5. Alternately, you can manually re-create each export on the CES cluster by using the **mmnfs** command.
   ```
   mmnfs export add exportPath --client client_options
   ```
6. Before you proceed to configure CES nodes, remove the NFS exports from `/etc/exports` from each of the old CNFS nodes
7. Add the IPs that were previously assigned to CNFS to the address pool to be managed by CNFS by using the following command:
   ```
   mmces address add —-node node1Name —-ces-ip ipAddress
   ```
   See "CES network configuration" on page 112 for details about how to use this command.
8. Take care to ensure that the IP addresses are unique and valid for your subnet

For more information about creating protocol data exports, see the *IBM Spectrum Scale: Administration and Programming Reference*.

Test access to new exports on CES NFS

Test and verify that you have the same level of access to the NFS exports as you did on CNFS to ensure that your applications and NFS clients can continue without further changes.

Chapter 9. Performance monitoring

With IBM Spectrum Scale, system administrators can monitor the performance of GPFS and the communications protocols that it uses.

Performance topics include:
- "Network performance monitoring"
- "Performance monitoring tool overview" on page 125

Network performance monitoring

Network performance can be monitored with Remote Procedure Call (RPC) statistics.

The GPFS daemon caches statistics relating to RPCs. Most statistics are related to RPCs sent to other nodes. This includes a set of up to seven statistics cached per node and one statistic that is cached per size of the RPC message. For RPCs received from other nodes, one statistic is cached for each type of RPC message.

The statistics cached per node are the following:

Channel wait time
The amount of time the RPC must wait for access to a communication channel to the target node.

Send time TCP
The amount of time to transfer an RPC message to an Ethernet interface.

Send time verbs
The amount of time to transfer an RPC message to an InfiniBand interface.

Receive time TCP
The amount of time to transfer an RPC message from an Ethernet interface into the daemon.

Latency TCP
The latency of the RPC when sent and received over an Ethernet interface.

Latency verbs
The latency of the RPC when sent and received over an InfiniBand interface.

Latency mixed
The latency of the RPC when sent over one type of interface (Ethernet or InfiniBand) and received over the other (InfiniBand or Ethernet).

If an InfiniBand network is not configured, no statistics are cached for send time verbs, latency verbs, and latency mixed.

The latency of an RPC is defined as the round-trip time minus the execution time on the target node. The round-trip time is measured from the start of writing the RPC message to the interface until the RPC reply is completely received. The execution time is measured on the target node from the time the message is completely received until the time the reply is sent. The latency, therefore, is the amount of time the RPC is being transmitted and received over the network and is a relative measure of the network performance as seen by the GPFS daemon.

There is a statistic associated with each of a set of size ranges, each with an upper bound that is a power of 2. The first range is 0 through 64, then 65 through 128, then 129 through 256, and then continuing until the last range has an upper bound of twice the **maxBlockSize**. For example, if the **maxBlockSize** is 1 MB, the upper bound of the last range is 2,097,152 (2 MB). For each of these ranges, the associated statistic is the latency of the RPC whose size falls within that range. The size of an RPC is the amount of data sent plus the amount of data received. However, if one amount is more than 16 times greater than the other, only the larger amount is used as the size of the RPC.

The final statistic associated with each type of RPC message, on the node where the RPC is received, is the execution time of the RPC.

Each of the statistics described so far is actually an aggregation of values. By default, an aggregation consists of 60 one-second intervals, 60 one-minute intervals, 24 one-hour intervals, and 30 one-day intervals. Each interval consists of a sum of values accumulated during the interval, a count of values added into the sum, the minimum value added into the sum, and the maximum value added into the sum. Sixty seconds after the daemon starts, each of the one-second intervals contains data and every second thereafter the oldest interval is discarded and a new one entered. An analogous pattern holds for the minute, hour, and day periods.

As each RPC reply is received, the following information is saved in a *raw statistics buffer*:
- channel wait time
- send time
- receive time
- latency
- length of data sent
- length of data received
- flags indicating if the RPC was sent or received over InfiniBand
- target node identifier

As each RPC completes execution, the execution time for the RPC and the message type of the RPC is saved in a *raw execution buffer*. Once per second these raw buffers are processed and the values are added to the appropriate aggregated statistic. For each value, the value is added to the statistic's sum, the count is incremented, and the value is compared to the minimum and maximum, which are adjusted as appropriate. Upon completion of this processing, for each statistic the sum, count, minimum, and maximum values are entered into the next one-second interval.

Every 60 seconds, the sums and counts in the 60 one-second intervals are added into a one-minute sum and count. The smallest of the 60 minimum values is determined, and the largest of the 60 maximum values is determined. This one-minute sum, count, minimum, and maximum are then entered into the next one-minute interval.

An analogous pattern holds for the minute, hour, and day periods. For any one particular interval, the sum is the sum of all raw values processed during that interval, the count is the count of all values during that interval, the minimum is the minimum of all values during that interval, and the maximum is the maximum of all values during that interval.

When statistics are displayed for any particular interval, an average is calculated from the sum and count, then the average, minimum, maximum, and count are displayed. The average, minimum and maximum are displayed in units of milliseconds, to three decimal places (one microsecond granularity).

The following **mmchconfig** attributes are available to control the RPC buffers and intervals:
- **rpcPerfRawStatBufferSize**
- **rpcPerfRawExecBufferSize**
- **rpcPerfNumberSecondIntervals**
- **rpcPerfNumberMinuteIntervals**
- **rpcPerfNumberHourIntervals**
- **rpcPerfNumberDayIntervals**

The **mmdiag** command with the **--rpc** parameter can be used to query RPC statistics.

For more information, see the topics *mmchconfigcommand* and *mmdiag command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Performance monitoring tool overview

The performance monitoring tool collects metrics from GPFS and protocols and provides performance information.

The performance monitoring system is started by default and consists of three parts: Collectors, Sensors, and Proxies.

Collector

In the previous release of IBM Spectrum Scale, the performance monitoring tool could be configured with a single collector only. From version 4.2, the performance monitoring tool can be configured with multiple collectors to increase scalability and fault-tolerance. This latter configuration is referred to as federation.

In a multi-collector federated configuration, the collectors need to be aware of each other, otherwise a collector would only return the data stored in its own measurement database. Once the collectors are aware of their peer collectors, they can collaborate with each other to collate measurement data for a given measurement query. All collectors that are part of the federation are specified in the peers configuration option in the collector's configuration file as shown in the following example:

```
peers = { host = "collector1.mydomain.com" port = "9085" },
    { host = "collector2.mydomain.com" port = "9085" }
```

The port number is the one specified by the `federationport` configuration option, typically set to 9085. You can also list the current host so that the same configuration file can be used for all the collector machines.

Once the peers have been specified, any query for measurement data might be directed to any of the collectors listed in the peers section and the collector collects and assembles a response based on all relevant data from all collectors. Hence, clients need to only contact a single collector instead of all of them in order to get all the measurements available in the system.

To distribute the measurement data reported by sensors over multiple collectors, multiple collectors might be specified when configuring the sensors.

If multiple collectors are specified, the sensors pick one to report their measurement data to. The sensors use stable hashes to pick the collector such that the sensor-collector relationship does not change too much if new collectors are added or if a collector is removed.

Additionally, sensors and collectors can be configured for high availability. In this setting, sensors report their measurement data to more than one collector such that the failure of a single collector would not lead to any data loss. For instance, if the collector redundancy is increased to two, every sensor reports to two collectors. As a side-effect of increasing the redundancy to two, the bandwidth consumed for reporting measurement data is duplicated. The collector redundancy has to be configured before the sensor configuration is stored in IBM Spectrum Scale by changing the **`colRedundancy`** option in `/opt/IBM/zimon/ZIMonSensors.cfg`.

Sensors

The sensors run on any node that is required to collect metrics. By default, the sensors are started on every node.

Sensors identify the collector from the information present in the configuration file, `/opt/IBM/zimon/ZIMonSensors.cfg`.

Proxies

A proxy is run for each of the protocols to collect the metrics for that protocol.

By default, the NFS and SMB proxies are started automatically with those protocols. They do not need to be started or stopped. However, to retrieve metrics for SMB, NFS or Object, these protocols have to be active on the specific node.

For information on enabling Object metrics, see the "Enabling Object metrics" on page 130 topic.

Configuring the performance monitoring tool

The performance monitoring tool, collector, sensors, and proxies, are a part of the IBM Spectrum Scale distribution. The tool is installed with the GPFS core packages on all nodes. The tools packages are small, approximately 300 KB for the sensors and 500 KB for the collector.

Note: The tool is supported on Linux nodes only.

For information on the usage of ports for the performance monitoring tool, see the "Firewall recommendations for Performance Monitoring tool" on page 413

Configuring sensors

The install toolkit sets up a default set of sensors to monitor on each node. You can modify the sensors on each individual node.

The configuration file of the sensors, `ZimonSensors.cfg`, is located on each node in the `/opt/IBM/zimon` folder. The file lists all groups of sensors in it. The configuration file includes the parameter setting of the sensors, such as the reporting frequency, and controls the sensors that are active within the cluster.

For example:

```
sensors =
{
        name = "CPU"
        period = 1
},
{       name = "Load"
        period = 1
},
{
        name = "Memory"
        period = 1
},
{
        name = "Network"
        period = 1
        filter = "eth*"
        # filters are currently ignored.
},
{
        name = "Netstat"
        period = 1
},
```

Starting with version 4.2 of the performance monitoring tool, sensors can be configured on nodes that are part of a IBM Spectrum Scale cluster through a IBM Spectrum Scale based configuration mechanism. However, this also requires the installation of IBM Spectrum Scale 4.2 on all the nodes where a sensor is running and where the sensors are to be configured. It also requires the entire cluster to be at least running IBM Spectrum Scale 4.1.1 and the execution of `mmchconfig release=LATEST` command.

The IBM Spectrum Scale based configuration method allows the sensor configuration to be stored as part of the IBM Spectrum Scale configuration. IBM Spectrum Scale-based configuration is only available for the sensor configuration files (`/opt/IBM/zimon/ZIMonSensors.cfg`) but not for the collector configuration files (`/opt/IBM/zimon/ZIMonCollector.cfg`). In this setup, the `/opt/IBM/zimon/ZIMonSensors.cfg` configuration file on each IBM Spectrum Scale node is maintained by IBM Spectrum Scale. As a result, the file must not be edited manually because whenever IBM Spectrum Scale needs to update a configuration parameter, the file is regenerated and any manual modifications are overwritten. Before using IBM Spectrum Scale-based configuration, an initial configuration needs to be stored within IBM Spectrum Scale. Storing this initial configuration is accomplished with the **mmperfmon config generate** command:

```
prompt# mmperfmon config generate \
--collectors collector1.domain.com,collector2.domain.com,...
```

Once the configuration file has been stored within IBM Spectrum Scale, it can be activated as follows.

```
prompt# mmchnode --perfmon –N nodeclass1,nodeclass2,...
```

Note: Any previously existing configuration file is overwritten.

To deactivate the performance monitoring tool, the same command is used but with the `--noperfmon` switch supplied instead. Configuration parameters can be changed with the following command where `param`*i* is of the form `sensorname.sensorattribute`:

```
prompt# mmperfmon config update param1=value1 param2=value2 ...
```

For instance, to restrict a sensor, say **NFSIO**, to a given node class and change the reporting period to once every 10 hours, one would specify `NFSIO.period=36000 NFSIO.restrict=nodeclass1` as attribute value pairs in the update command.

Note: Some sensors such as the cluster export services sensors must only run on a specific set of nodes. Other sensors such as the `GPFSDiskCap` sensor must only run on a single node in the cluster since the data reported is the same independently of the node the sensor is running on. For these types of sensors, the restrict function is especially intended.

Configuration information for SMB and NFS in the `ZimonSensors.cfg` file references the sensor definition files in the `/opt/IBM/zimon` folder. For example:

- The `CTDBDBStats.cfg` file is referred in:

  ```
  {       name = "CTDBDBStats"
          period = 1
          type = "Generic"
  },
  ```

- The `CTDBStats.cfg` file is referred in:

  ```
  {       name = "CTDBStats"
          period = 1
          type = "Generic"
  },
  ```

- The `NFSIO.cfg` file is referred in:

  ```
  {
          # NSF Ganesha statistics
          name = "NFSIO"
          period = 1
          type = "Generic"
  },
  ```

- The `SMBGlobalStats.cfg` file is referred in:

```
{       name = "SMBGlobalStats"
        period = 1
        type = "Generic"
},
```

- The SMBStats.cfg file is referred in:

```
{       name = "SMBStats"
        period = 1
        type = "Generic"
},
```

The period in the example specifies the interval size in number of seconds when a sensor group will gather data. 0 means the sensor group is disabled and 1 runs the sensor group every second. You can specify a higher value to decrease the frequency that the data is collected.

The file also contains the hostname of the node where the collector is running that the sensor should be reporting to.

Configuration changes result in a new version of the configuration file which is then propagated through the IBM Spectrum Scale cluster at the file level.

Note: Some sensors, such as VFS, are not enabled by default even though they have associated predefined queries with the **mmperfmon query** command because the collector may display performance issues of its own if it is required to collect more than 1000000 metrics per second.

To enable VFS sensors, use the **mmfsadm vfsstats enable** command on the node.
To enable a sensor, set the period value to an integer greater than 0. You will then need to restart the sensors on that node using the **systemctl restart pmsensors** command:

The sensors can be stopped (deactivated) using the **systemctl stop pmsensors** command.

The sensors can be started (activated) enabled using the **systemctl start pmsensors** command.

Configuring the collector

The configuration file of the collector, ZIMonCollector.cfg, is located in the /opt/IBM/zimon/ folder.

The most important configuration options are the domains and the peers configuration options. All other configuration options are best left at their defaults and are explained within the default configuration file shipped with ZIMon.

Metric Domain Configuration

The domains configuration indicates the number of metrics to be collected and how long they must be retained and in what granularity. Multiple domains might be specified. If data no longer fits into the current domain, data is spilled over into the next domain and resampled.

A simple configuration is:

```
domains = {
# this is the raw domain, aggregation factor for the raw domain is always 0
aggregation = 0
ram = "500m" # amount of RAM to be used
duration = "12h"
filesize = "1g" # maximum file size
files = 16 # number of files.
}
```

,

```
{
# this is the second domain that aggregates to 60 seconds
aggregation = 60
ram = "500m" # amount of RAM to be used
duration = "4w"
filesize = "500m" # maximum file size
files = 4 # number of files.
}


,
{
# this is the third domain that aggregates to 30*60 seconds == 30 minutes
aggregation = 30
ram = "500m" # amount of RAM to be used
duration = "1y"
filesize = "500m" # maximum file size
files = 4 # number of files.
}
```

The configuration file lists several data domains. At least one domain must be present and the first domain represents the raw data collection as the data is collected by sensors. The aggregation parameter for this first domain must be set to 0.

Each domain specifies the following parameters:

- The **duration** parameter indicates how long until the collected metrics will be pushed into the next (coarser-grained) domain. If this option is left out, no limit on the duration will be imposed.
- The **ram** parameter indicates the amount of RAM to be allocated for the domain. Once that amount of RAM has been filled up, collected metrics will be pushed into the next (coarser-grained) domain. If this option is left out, no limit on the amount of RAM available will be imposed.
- The **filesize** and **files** parameter indicates how much space is allocated on disk for a given domain. While storing metrics in memory, there is a persistence mechanism in place that also stores the metrics on disk in files of size filesize. Once the number of files is reached and a new file is to be allocated, the oldest file is removed from the disk. The persistent storage must be at least as large as the amount of main memory to be allocated for a domain because when the collector is restarted, the in-memory database is recreated from these files. Queries can also be served from these files if they are executed in archive mode (`-a`).

The aggregation value, used for the second and following domains, indicates the resampling to be performed. Once data is spilled into this domain, the data is resampled to be no better than indicated by the aggregation factor. The value for the second domain is in seconds, the value for domain n (n>2) is the value of domain n-1 multiplied by the aggregation value of domain n.

The collector collects the metrics from the sensors. For instance, in a 5 node cluster where only the load values (load1, load5, load15) are reported, the collector will have to maintain 15 metrics (3 metrics times 5 nodes). Depending on the number of metrics collected, the collector requires a different amount of main memory to store the collected metrics in memory. Assuming 500000 metrics are collected, here are two configurations and the amount of data required to store the database. Depending on the amount of data to be collected, 500000 metrics corresponds to about 1000 nodes.

Configuration 1 (4GB of RAM). Domain one configured at 1 second granularity for a period of 6 hours, domain 2 configured at 30 seconds granularity for the next 2 days, domain 3 configured at 15 minutes granularity for the next 2 weeks and domain 4 configured at 6 hour granularity for the next 2 months.

Configuration 2 (16GB of RAM). Domain one configured at 1 second granularity for a period of 1 day, domain 2 configured at 30 sec granularity for the next week, domain 3 configured at 15 minute granularity for the next 2 months and domain 4 configured at 6 hour granularity for the next year.

Note: The above computation only gives the memory required for the in-memory database, not including the indices necessary for the persistent storage or for the collector program itself.

The collectors can be stopped (deactivated) using the **systemctl stop pmcollector** command.

The collectors can be started (activated) using the **systemctl start pmcollector** command.

Manual configuration method

When upgrading the performance monitoring tool, it is important to note how the previous version was configured and if the configuration mechanism is to be changed. Before IBM Spectrum Scale 4.2 the previous version was configured using a file-based configuration mechanism where you had to manually edit the configuration files and propagate them to the requisite nodes. If the configuration mechanism is to be changed, it is important to verify that the installed versions of both IBM Spectrum Scale and the performance monitoring tool support the new configuration mechanism. If this is not the case, or if you would use the manual, file-based, configuration method, then when you install the IBM Spectrum Scale 4.2 there are a couple of steps required. None of the nodes in the cluster should be designated **perfmon** nodes. If the nodes in the cluster are designated as **perfmon** nodes then you must run `mmchnode --perfmon –N all` command.

Then you need to delete the centrally stored configuration information by issuing `mmperfmon config delete --all` command.

The `/opt/IBM/zimon/ZIMonSensors.cfg` file is then maintained manually and the tools using it can overwrite it with a new version at any time. This mode is useful if sensors are to be installed on non-Spectrum Scale nodes or if you want to have a cluster with multiple levels of IBM Spectrum Scale running.

Enabling Object metrics

At the time of installation, the Object metrics proxy is configured to start by default on an Object protocol node.

The Object metrics proxy server is controlled by the corresponding service script called `pmswiftd`, located in the `/etc/rc.d/init.d/` directory. You can start and stop it using the **systemctl start pmswiftd** and **systemctl stop pmswiftd** commands respectively.

In case of a system restart, the Object metrics proxy server restarts automatically whereas the Object metrics proxy client is triggered by the Performance Monitoring tool. In case of a failover, the server may start automatically if it has failed gracefully. Otherwise, it needs to be started manually.

Restarting the performance monitoring tool

You must manually restart the performance monitoring tool after you install a later version or reconfigure it.

Use the **systemctl start pmsensors** command to start monitoring on that node.

Use the **systemctl start pmcollector** command on the node running the collector.

When restarting the tool, the sensors identify the collector from the information given in the `ZimonSensors.cfg` file that is updated on every node.

The Performance monitoring tool also needs to restart if you make changes in the configuration files of either sensors, `ZimonSensors.cfg`, or the collector, `ZIMonCollector.cfg`, located in the `/opt/IBM/zimon/` folder.

To restart the sensor on a particular node, use the **systemctl restart pmsensors** command.

To restart the collector, use the **systemctl restart pmcollector** command.

Defining the metrics

For performance reasons, the performance monitoring tool by default does not collect all the available metrics. You can add other metrics to focus on particular performance problems.

For the available metrics, see "List of metrics" on page 136.

After you define the required metrics in the configuration file, you must restart all the sensors. To restart the sensors, run the following command:

```
systemctl restart pmsensors
```

Performance monitoring

To view the metrics associated with GPFS and the associated protocols, run the **mmperfmon** command the **query** option.

You can also use the **mmperfmon**command with the **query** option to detect performance issues and problems. You can collect metrics for all nodes or for a particular node.

For more information, see the topic *mmperfmoncommand* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Analyzing performance issues

This section describes how to use the **mmperfmon** with the **query** option to investigate various types of problems.

- Problem: System slowing down

  Use **mmperfmon query compareNodes cpu_user** or **mmperfmon query compareNodes cpu_system** command to compare CPU metrics for all the nodes in your system.

  1. Is there a node that has a significantly higher CPU utilization for the entire time period? Does this continue? You might need to need to investigate further on this node.
  2. Is there a node that has significantly lower CPU utilization over the entire period? Does this node have a health problem?
  3. Use **mmperfmon query compareNodes protocolThroughput** to look at the throughput for each of the nodes for the different protocols.

     Note: Note that the metrics of each individual protocol will not always include exact I/O figures.
  4. Use **mmperfmon query compareNodes protocolIORate** to look at the IO performance for each of the nodes in your system.
- Problem: A particular node is causing problems

  Use **mmperfmon query usage** to show the CPU, memory, storage, and network usage.
- Problem: A particular protocol is causing problems

  Use **mmperfmon query** to investigate problems with your specific protocol. You can compare cross-node metrics using **mmperfmon query compareNodes**.

  For example, **mmperfmon query compareNodes nfs_read_ops**

  Compare the NFS read operations on all the nodes that are using NFS. By comparing the different NFS metrics, you will be able to identify which node is causing the problems. The problem will either manifest itself as running with much higher values than the other nodes, or much lower (depending on the issue) when considered over several buckets of time.
- Problem: A particular protocol is causing problems on a particular node.

Use **mmperfmon query** on the particular node to look deeper into the protocol performance on that node.

For example, if there is a problem with NFS:

– **mmperfmon query nfsIOlatency** - To get details of the nfsIOlatency,
– **mmperfmon query nfsIOrate** - To get details of the NFS IO rate.
– **mmperfmon query nfsThroughput** - To get details of the NFS throughput

List of queries

You can make the following predfined queries with **query** option of the **mmperfmon**command.

General and network

- **usage**: Retrieves details about the CPU, memory, storage and network usage
- **cpu**: Retrieves details of the CPU utilization in system and user space, and context switches.
- **netDetails**: Retrieves details about the network.
- **NetErrors**: Retrieves details about network problems, such as collisions, drops, and errors, for all available networks.
- **compareNodes**: Compares a single metric across all nodes running sensors

GPFS

GPFS metric queries gives an overall view of the GPFS without considering the protocols.

- **gpfsCRUDopsLatency**: Retrieves information about the GPFS CRUD operations latency
- **gpfsFSWaits**: Retrieves information on the maximum waits for read and write operations for all file systems.
- **gpfsNSDWaits**: Retrieves information on the maximum waits for read and write operations for all disks.
- **gpfsNumberOperations**: Retrieves the number of operations to the GPFS file system.
- **gpfsVFSOpCounts**: Retrieves VFS operation counts.

Cross protocol

These queries retrieve information after comparing metrics between different protocols on a particular node.

- **protocolIOLatency**: Compares latency per protocol (SMB, NFS, Object).
- **protocolIORate**: Retrieves the percentage of total I/O rate per protocol (SMB, NFS, Object).
- **protocolThroughput**: Retrieves the percentage of total throughput per protocol (SMB, NFS, Object).

NFS

These queries retrieve metrics associated with the NFS protocol.

- **nfsIOLatency**: Retrieves the NFS IO Latency in nanoseconds.
- **nfsIORate**: Retrieves the NFS IOps per second.
- **nfsThroughput**: Retrieves the NFS Throughput in bytes per second.
- **nfsErrors**: Retrieves the NFS error count for read and write operations.
- **nfsQueue**: Retrieves the NFS read and write queue latency in nanoseconds.
- **nfsThroughputPerOp**: Retrieves the NFS read and write throughput per op in bytes

Object

- **objAcc**: Details on the Object Account performance

  Retrieved metrics:

  - **account_auditor_time**
  - **account_reaper_time**
  - **account_replicator_time**
  - **account_DEL_time**
  - **account_DEL_err_time**
  - **account_GET_time**
  - **account_GET_err_time**
  - **account_HEAD_time**
  - **account_HEAD_err_time**
  - **account_POST_time**
  - **account_POST_err_time**
  - **account_PUT_time**
  - **account_PUT_err_time**
  - **account_REPLICATE_time**
  - **account_REPLICATE_err_time**
- **objCon**: Details on the Object Container performance

  Retrieved metrics:
  - **container_auditor_time**
  - **container_replicator_time**
  - **container_DEL_time**
  - **container_DEL_err_time**
  - **container_GET_time**
  - **container_GET_err_time**
  - **container_HEAD_time**
  - **container_HEAD_err_time**
  - **container_POST_time**
  - **container_POST_err_time**
  - **container_PUT_time**
  - **container_PUT_err_time**
  - **container_REPLICATE_time**
  - **container_REPLICATE_err_time**
  - **container_sync_deletes_time**
  - **container_sync_puts_time**
  - **container_updater_time**
- **objObj**: Details on the Object performance

  Retrieved metrics:
  - **object_auditor_time**
  - **object_expirer_time**
  - **object_replicator_partition_delete_time**
  - **object_replicator_partition_update_time**
  - **object_DEL_time**
  - **object_DEL_err_time**
  - **object_GET_time**
  - **object_GET_err_time**
  - **object_HEAD_time**

  - **object_HEAD_err_time**
  - **object_POST_time**
  - **object_POST_err_time**
  - **object_PUT_time**
  - **object_PUT_err_time**
  - **object_REPLICATE_err_time**
  - **object_REPLICATE_time**
  - **object_updater_time**
- **objPro**: Details on the Object Proxy performance

  Retrieved metrics:
  - **proxy_account_latency**
  - **proxy_container_latency**
  - **proxy_object_latency**
  - **proxy_account_GET_time**
  - **proxy_account_GET_bytes**
  - **proxy_account_HEAD_time**
  - **proxy_account_HEAD_bytes**
  - **proxy_container_DEL_time**
  - **proxy_container_DEL_bytes**
  - **proxy_container_GET_time**
  - **proxy_container_GET_bytes**
  - **proxy_container_HEAD_time**
  - **proxy_container_HEAD_bytes**
  - **proxy_container_PUT_time**
  - **proxy_container_PUT_bytes**
  - **proxy_object_DEL_time**
  - **proxy_object_DEL_bytes**
  - **proxy_object_GET_time**
  - **proxy_object_GET_bytes**
  - **proxy_object_HEAD_time**
  - **proxy_object_HEAD_bytes**
  - **proxy_object_PUT_time**
  - **proxy_object_PUT_bytes**
- **objAccIO**: Information on the Object Account IO rate

  Retrieved metrics:
  - **account_GET_time**
  - **account_GET_err_time**
  - **account_HEAD_time**
  - **account_HEAD_err_time**
  - **account_POST_time**
  - **account_POST_err_time**
  - **account_PUT_time**
  - **account_PUT_err_time**
- **objConIO**: Information on the Object Container IO rate

  Retrieved metrics:

  - **container_GET_time**
  - **container_GET_err_time**
  - **container_HEAD_time**
  - **container_HEAD_err_time**
  - **container_POST_time**
  - **container_POST_err_time**
  - **container_PUT_time**
  - **container_PUT_err_time**
- **objObjIO**: Information on the Object Object IO rate

  Retrieved metrics:
  - **object_GET_time**
  - **object_GET_err_time**
  - **object_HEAD_time**
  - **object_HEAD_err_time**
  - **object_POST_time**
  - **object_POST_err_time**
  - **object_PUT_time**
  - **object_PUT_err_time**
- **objProIO**: Information on the Object Proxy IO rate

  Retrieved metrics:
  - **proxy_account_GET_time**
  - **proxy_account_GET_bytes**
  - **proxy_container_GET_time**
  - **proxy_container_GET_bytes**
  - **proxy_container_PUT_time**
  - **proxy_container_PUT_bytes**
  - **proxy_object_GET_time**
  - **proxy_object_GET_bytes**
  - **proxy_object_PUT_time**
  - **proxy_object_PUT_bytes**
- **objAccThroughput**: Information on the Object Account Throughput

  Retrieved metrics:
  - **account_GET_time**
  - **account_PUT_time**
- **objConThroughput**: Information on the Object Container Throughput

  Retrieved metrics:
  - **container_GET_time**
  - **container_PUT_time**
- **objObjThroughput**: Information on the Object Throughput

  Retrieved metrics:
  - **object_GET_time**
  - **object_PUT_time**
  -
- **objProThroughput**: Information on the Object Proxy Throughput

  Retrieved metrics:

- **`proxy_account_GET_time`**
  - **`proxy_account_GET_bytes`**
  - **`proxy_container_GET_time`**
  - **`proxy_container_GET_bytes`**
  - **`proxy_container_PUT_time`**
  - **`proxy_container_PUT_bytes`**
  - **`proxy_object_GET_time`**
  - **`proxy_object_GET_bytes`**
  - **`proxy_object_PUT_time`**
  - **`proxy_object_PUT_bytes`**
- **`objAccLatency`**: Information on the Object Account Latency

  Retrieved metric:
  - **`proxy_account_latency`**
- **`objConLatency`**: Information on the Object Container Latency

  Retrieved metric:
  - **`proxy_container_latency`**
- **`objObjLatency`**: Information on the Object Latency

  Retrieved metric:
  - **`proxy_object_latency`**

SMB

These queries retrieve metrics associated with SMB.
- **`smb2IOLatency`**: Retrieves the SMB2 I/O latencies per bucket size (default 1 sec).
- **`smb2IORate`**: Retrieves the SMB2 I/O rate in number of operations per bucket size (default 1 sec).
- **`smb2Throughput`**: Retrieves the SMB2 Throughput in bytes per bucket size (default 1 sec).
- **`smb2Writes`** : Retrieves count, # of idle calls, bytes in and out, and operation time for SMB2 writes.
- **`smbConnections`**: - Retrieves the number of SMB connections.

CTDB

These queries retrieve metrics associated with CTDB.
- **`ctdbCallLatency`**: Retrieves information on the CTDB call latency.
- **`ctdbHopCountDetails`**: Retrieves information on the CTDB hop count buckets 0 to 5 for one database.
- **`ctdbHopCounts`** :Retrieves information on the CTDB hop counts (bucket 00 = 1-3 hops) for all databases.

List of metrics

The performance monitoring tool can report the following metrics:

Network and general

All network and general metrics are native. There are no computed metrics in this section.

CPU

This section lists information about CPU in the system
- **cpu_contexts**: Number of context switches across all CPU cores.
- **cpu_guest**: Percentage of total CPU spent running a guest OS. Included in cpu_user.

- **cpu_guest_nice**: Percentage of total CPU spent running as niced guest OS. Included in cpu_nice.
- **cpu_hiq**: Percentage of total CPU spent serving hardware interrupts.
- **cpu_idle**: Percentage of total CPU spent idling.
- **cpu_interrupts**: Number of interrupts serviced.
- **cpu_iowait**: Percentage of total CPU spent waiting for I/O to complete.
- **cpu_nice**: Percentage of total CPU time spent in lowest-priority user processes.
- **cpu_siq**: Percentage of total CPU spent serving software interrupts.
- **cpu_steal**: Percentage of total CPU spent waiting for other OS when running in a virtualized environment.
- **cpu_system**: Percentage of total CPU time spent in kernel mode.
- **cpu_user**: Percentage of total CPU time spent in normal priority user processes.

DiskFree

This section contains information about the free disk. Each mounted directory will have a separate section, for example `DiskFree|/boot/df_free`
- **df_free**: Amount of free disk space on the file system
- **df_total**: Amount of total disk space on the file system
- **df_used**: Amount of used disk space on the file system

Diskstat

This section contains Disk status information for each of the disks. For example, `Diskstat|sda|disk_active_ios`.
- **disk_active_ios**: Number of I/O operations currently in progress.
- **disk_aveq**: Weighted number of milliseconds spent doing I/Os.
- **disk_io_time**: Number of milliseconds the system spent doing I/O operation.
- **disk_read_ios**: Total number of read operations completed successfully.
- **disk_read_merged**: Number of (small) read operations that have been merged into a larger read.
- **disk_read_sect**: Number of sectors read.
- **disk_read_time**: Amount of time in milliseconds spent reading.
- **disk_write_ios**: Number of write operations completed successfully.
- **disk_write_merged**: Number of (small) write operations that have been merged into a larger write.
- **disk_write_sect**: Number of sectors written.
- **disk_write_time**: Amount of time in milliseconds spent writing.

Load
- **jobs**: The total number of jobs that currently exist in the system.
- **load1**: The average load (number of jobs in the run queue) over the last minute.
- **load15**: The average load (number of jobs in the run queue) over the last 15 minutes.
- **load5**: The average load (number of jobs in the run queue) over the five minutes.

Memory
- **mem_active**: Active memory that was recently accessed.
- **mem_active_anon**: Active memory with no file association, that is, heap and stack memory.
- **mem_active_file**: Active memory that is associated with a file, for example, page cache memory.
- **mem_buffers**: Temporary storage used for raw disk blocks.

- **mem_cached**: In-memory cache for files read from disk (the page cache). Does not include mem_swapcached.
- **mem_dirty**: Memory which is waiting to get written back to the disk.
- **mem_inactive**: Inactive memory that hasn't been accessed recently.
- **mem_inactive_anon**: Inactive memory with no file association, that is, inactive heap and stack memory.
- **mem_inactive_file**: Inactive memory that is associated with a file, for example, page cache memory.
- **mem_memfree**: Total free RAM.
- **mem_memtotal**: Total usable RAM.
- **mem_mlocked**: Memory that is locked.
- **mem_swapcached**: In-memory cache for pages that are swapped back in.
- **mem_swapfree**: Amount of swap space that is currently unused.
- **mem_swaptotal**: Total amount of swap space available.
- **mem_unevictable**: Memory that cannot be paged out.

Netstat
- **ns_closewait**: Number of connections in state TCP_CLOSE_WAIT
- **ns_established**: Number of connections in state TCP_ESTABLISHED
- **ns_listen**: Number of connections in state TCP_LISTEN
- **ns_local_bytes_r**: Number of bytes received (local -> local)
- **ns_local_bytes_s**: Number of bytes sent (local -> local)
- **ns_localconn**: Number of local connections (local -> local)
- **ns_remote_bytes_r**: Number of bytes sent (local -> remote)
- **ns_remote_bytes_s**: Number of bytes sent (remote -> local)
- **ns_remoteconn**: Number of remote connections (local -> remote)
- **ns_timewait**: Number of connections in state TCP_TIME_WAIT

Network
- **netdev_bytes_r**: Number of bytes received.
- **netdev_bytes_s**: Number of bytes sent.
- **netdev_carrier**: Number of carrier loss events.
- **netdev_collisions**: Number of collisions.
- **netdev_compressed_r**: Number of compressed frames received.
- **netdev_compressed_s**: Number of compressed packets sent.
- **netdev_drops_r**: Number of packets dropped while receiving.
- **netdev_drops_s**: Number of packets dropped while sending.
- **netdev_errors_r**: Number of read errors.
- **netdev_errors_s**: Number of write errors.
- **netdev_fifo_r**: Number of FIFO buffer errors.
- **netdev_fifo_s**: Number of FIFO buffer errors while sending.
- **netdev_frames_r**: Number of frame errors while receiving.
- **netdev_multicast_r**: Number of multicast packets received.
- **netdev_packets_r**: Number of packets received.
- **netdev_packets_s**: Number of packets sent.

GPFS

GPFSDisk

For each NSD in the system, for example `GPFSDisk|myMachine|myFilesystem|myNSD|gpfs_ds_bytes_read`

- **gpfs_ds_bytes_read**: Number of bytes read.
- **gpfs_ds_bytes_written**: Number of bytes written.
- **gpfs_ds_max_disk_wait_rd**: The longest time spent waiting for a disk read operation.
- **gpfs_ds_max_disk_wait_wr**: The longest time spent waiting for a disk write operation.
- **gpfs_ds_max_queue_wait_rd**: The longest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_ds_max_queue_wait_wr**: The longest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_ds_min_disk_wait_rd**: The shortest time spent waiting for a disk read operation.
- **gpfs_ds_min_disk_wait_wr**: The shortest time spent waiting for a disk write operation.
- **gpfs_ds_min_queue_wait_rd**: The shortest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_ds_min_queue_wait_wr**: The shortest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_ds_read_ops**: Number of read operations.
- **gpfs_ds_tot_disk_wait_rd**: The total time in seconds spent waiting for disk read operations.
- **gpfs_ds_tot_disk_wait_wr**: The total time in seconds spent waiting for disk write operations.
- **gpfs_ds_tot_queue_wait_rd**: The total time spent between being enqueued for a read operation and the completion of that operation.
- **gpfs_ds_tot_queue_wait_wr**: The total time spent between being enqueued for a write operation and the completion of that operation.
- **gpfs_ds_write_ops**: Number of write operations.

GPFSFileSystem

For each file system, for example `GPFSFileSystem`

- **gpfs_fs_bytes_read**: Number of bytes read.
- **gpfs_fs_bytes_written**: Number of bytes written.
- **gpfs_fs_disks**: Number of disks in the file system.
- **gpfs_fs_max_disk_wait_rd**: The longest time spent waiting for a disk read operation.
- **gpfs_fs_max_disk_wait_wr**: The longest time spent waiting for a disk write operation.
- **gpfs_fs_max_queue_wait_rd**: The longest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_fs_max_queue_wait_wr**: The longest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_fs_min_disk_wait_rd**: The shortest time spent waiting for a disk read operation.
- **gpfs_fs_min_disk_wait_wr**: The shortest time spent waiting for a disk write operation.
- **gpfs_fs_min_queue_wait_rd**: The shortest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_fs_min_queue_wait_wr**: The shortest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_fs_read_ops**: Number of read operations
- **gpfs_fs_tot_disk_wait_rd**: The total time in seconds spent waiting for disk read operations.

- **gpfs_fs_tot_disk_wait_wr**: The total time in seconds spent waiting for disk write operations.
- **gpfs_fs_tot_queue_wait_rd**: The total time spent between being enqueued for a read operation and the completion of that operation.
- **gpfs_fs_tot_queue_wait_wr**: The total time spent between being enqueued for a write operation and the completion of that operation.
- **gpfs_fs_write_ops**: Number of write operations.

GPFSFileSystemAPI
- **gpfs_fis_bytes_read**: Number of bytes read.
- **gpfs_fis_bytes_written**: Number of bytes written.
- **gpfs_fis_close_calls**: Number of close calls.
- **gpfs_fis_disks**: Number of disks in the file system.
- **gpfs_fis_inodes_written**: Number of inode updates to disk.
- **gpfs_fis_open_calls**: Number of open calls.
- **gpfs_fis_read_calls**: Number of read calls.
- **gpfs_fis_readdir_calls**: Number of readdir calls.
- **gpfs_fis_write_calls**: Number of write calls.

GPFSNSDDisk
- **gpfs_nsdds_bytes_read**: Number of bytes read.
- **gpfs_nsdds_bytes_written**: Number of bytes written.
- **gpfs_nsdds_max_disk_wait_rd**: The longest time spent waiting for a disk read operation.
- **gpfs_nsdds_max_disk_wait_wr**: The longest time spent waiting for a disk write operation.
- **gpfs_nsdds_max_queue_wait_rd**: The longest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_nsdds_max_queue_wait_wr**: The longest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_nsdds_min_disk_wait_rd**: The shortest time spent waiting for a disk read operation.
- **gpfs_nsdds_min_disk_wait_wr**: The shortest time spent waiting for a disk write operation.
- **gpfs_nsdds_min_queue_wait_rd**: The shortest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_nsdds_min_queue_wait_wr**: The shortest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_nsdds_read_ops**: Number of read operations.
- **gpfs_nsdds_tot_disk_wait_rd**: The total time in seconds spent waiting for disk read operations.
- **gpfs_nsdds_tot_disk_wait_wr**: The total time in seconds spent waiting for disk write operations.
- **gpfs_nsdds_tot_queue_wait_rd**: The total time spent between being enqueued for a read operation and the completion of that operation.
- **gpfs_nsdds_tot_queue_wait_wr**: The total time spent between being enqueued for a write operation and the completion of that operation.
- **gpfs_nsdds_write_ops**: Number of write operations.

GPFSNSDFS
- **gpfs_nsdfs_bytes_read**: Number of NSD bytes read, aggregated to the file system.
- **gpfs_nsdfs_bytes_written**: Number of NSD bytes written, aggregated to the file system.
- **gpfs_nsdfs_read_ops**: Number of NSD read operations, aggregated to the file system.
- **gpfs_nsdfs_write_ops**: Number of NSD write operations, aggregated to the file system.

GPFSNSDPool

- **gpfs_nsdpool_bytes_read**: Number of NSD bytes read, aggregated to the file system.
- **gpfs_nsdpool_bytes_written**: Number of NSD bytes written, aggregated to the file system.
- **gpfs_nsdpool_read_ops**: Number of NSD read operations, aggregated to the file system.
- **gpfs_nsdpool_write_ops**: Number of NSD write operations, aggregated to the file system.

GPFSNode

- **gpfs_ns_bytes_read**: Number of bytes read.
- **gpfs_ns_bytes_written**: Number of bytes written.
- **gpfs_ns_clusters**: Number of clusters participating
- **gpfs_ns_disks**: Number of disks in all mounted file systems
- **gpfs_ns_filesys**: Number of mounted file systems
- **gpfs_ns_max_disk_wait_rd**: The longest time spent waiting for a disk read operation.
- **gpfs_ns_max_disk_wait_wr**: The longest time spent waiting for a disk write operation.
- **gpfs_ns_max_queue_wait_rd**: The longest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_ns_max_queue_wait_wr**: The longest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_ns_min_disk_wait_rd**: The shortest time spent waiting for a disk read operation.
- **gpfs_ns_min_disk_wait_wr**: The shortest time spent waiting for a disk write operation.
- **gpfs_ns_min_queue_wait_rd**: The shortest time between being enqueued for a disk read operation and the completion of that operation.
- **gpfs_ns_min_queue_wait_wr**: The shortest time between being enqueued for a disk write operation and the completion of that operation.
- **gpfs_ns_read_ops**: Number of read operations.
- **gpfs_ns_tot_disk_wait_rd**: The total time in seconds spent waiting for disk read operations.
- **gpfs_ns_tot_disk_wait_wr**: The total time in seconds spent waiting for disk write operations.
- **gpfs_ns_tot_queue_wait_rd**: The total time spent between being enqueued for a read operation and the completion of that operation.
- **gpfs_ns_tot_queue_wait_wr**: The total time spent between being enqueued for a write operation and the completion of that operation.
- **gpfs_ns_write_ops**: Number of write operations.

GPFSNodeAPI

- **gpfs_is_bytes_read**: Number of bytes read.
- **gpfs_is_bytes_written**: Number of bytes written.
- **gpfs_is_close_calls**: Number of close calls.
- **gpfs_is_inodes_written**: Number of inode updates to disk.
- **gpfs_is_open_calls**: Number of open calls.
- **gpfs_is_readDir_calls**: Number of readdir calls.
- **gpfs_is_read_calls**: Number of read calls.
- **gpfs_is_write_calls**: Number of write calls.

GPFSPoolIO

For the system within each GPFS device:

- **gpfs_pool_bytes_rd**: Total size of all disks for this usage type.
- **gpfs_pool_bytes_wr**: Total available disk space in full blocks for this usage type.

- **gpfs_pool_free_fragkb**: Total available space in fragments for this usage type.

GPFSVFS

- **gpfs_vfs_accesses**: Number of accesses operations.
- **gpfs_vfs_accesses_t**: Amount of time in seconds spent in accesses operations.
- **gpfs_vfs_aioread**: Number of aioread operations.
- **gpfs_vfs_aioread_t**: Amount of time in seconds spent in aioread operations.
- **gpfs_vfs_aiowrite**: Number of aiowrite operations.
- **gpfs_vfs_aiowrite_t**: Amount of time in seconds spent in aiowrite operations.
- **gpfs_vfs_clear**: Number of clear operations.
- **gpfs_vfs_clear_t**: Amount of time in seconds spent in clear operations.
- **gpfs_vfs_close**: Number of close operations.
- **gpfs_vfs_close_t**: Amount of time in seconds spent in close operations.
- **gpfs_vfs_create**: Number of create operations.
- **gpfs_vfs_create_t**: Amount of time in seconds spent in create operations.
- **gpfs_vfs_decodeFh**: Number of decodeFh operations.
- **gpfs_vfs_decodeFh_t**: Amount of time in seconds spent in decodeFh operations.
- **gpfs_vfs_detDentry**: Number of detDentry operations.
- **gpfs_vfs_encodeFh**: Number of encodeFh operations.
- **gpfs_vfs_encodeFh_t**: Amount of time in seconds spent in encodeFh operations.
- **gpfs_vfs_flock**: Number of flock operations.
- **gpfs_vfs_flock_t**: Amount of time in seconds spent in flock operations.
- **gpfs_vfs_fsync**: Number of fsync operations.
- **gpfs_vfs_fsyncRange**: Number of fsyncRange operations.
- **gpfs_vfs_fsyncRange_t**: Amount of time in seconds spent in fsyncRange operations.
- **gpfs_vfs_fsync_t**: Amount of time in seconds spent in fsync operations.
- **gpfs_vfs_ftrunc**: Number of ftrunc operations.
- **gpfs_vfs_ftrunc_t**: Amount of time in seconds spent in ftrunc operations.
- **gpfs_vfs_getDentry_t**: Amount of time in seconds spent in getDentry operations.
- **gpfs_vfs_getParent**: Number of getParent operations.
- **gpfs_vfs_getParent_t**: Amount of time in seconds spent in getParent operations.
- **gpfs_vfs_getattr**: Number of getattr operations.
- **gpfs_vfs_getattr_t**: Amount of time in seconds spent in getattr operations.
- **gpfs_vfs_getxattr**: Number of getxattr operations.
- **gpfs_vfs_getxattr_t**: Amount of time in seconds spent in getxattr operations.
- **gpfs_vfs_link**: Number of link operations.
- **gpfs_vfs_link_t**: Amount of time in seconds spent in link operations.
- **gpfs_vfs_listxattr**: Number of listxattr operations.
- **gpfs_vfs_listxattr_t**: Amount of time in seconds spent in listxattr operations.
- **gpfs_vfs_lockctl**: Number of lockctl operations.
- **gpfs_vfs_lockctl_t**: Amount of time in seconds spent in lockctl operations.
- **gpfs_vfs_lookup**: Number of lookup operations.
- **gpfs_vfs_lookup_t**: Amount of time in seconds spent in lookup operations.
- **gpfs_vfs_mapLloff**: Number of mapLloff operations.
- **gpfs_vfs_mapLloff_t**: Amount of time in seconds spent in mapLloff operations.

- **gpfs_vfs_mkdir**: Number of mkdir operations.
- **gpfs_vfs_mkdir_t**: Amount of time in seconds spent in mkdir operations.
- **gpfs_vfs_mknod**: Number of mknod operations.
- **gpfs_vfs_mknod_t**: Amount of time in seconds spent in mknod operations.
- **gpfs_vfs_mmapread**: Number of mmapread operations.
- **gpfs_vfs_mmapread_t**: Amount of time in seconds spent in mmapread operations.
- **gpfs_vfs_mmapwrite**: Number of mmapwrite operations.
- **gpfs_vfs_mmapwrite_t**: Amount of time in seconds spent in mmapwrite operation.
- **gpfs_vfs_mount**: Number of mount operations.
- **gpfs_vfs_mount_t**: Amount of time in seconds spent in mount operations.
- **gpfs_vfs_open**: Number of open operations.
- **gpfs_vfs_open_t**: Amount of time in seconds spent in open operations.
- **gpfs_vfs_read**: Number of read operations.
- **gpfs_vfs_read_t**: Amount of time in seconds spent in read operations.
- **gpfs_vfs_readdir**: Number of readdir operations.
- **gpfs_vfs_readdir_t**: Amount of time in seconds spent in readdir operations.
- **gpfs_vfs_readlink**: Number of readlink operations.
- **gpfs_vfs_readlink_t**: Amount of time in seconds s.pent in readlink operations
- **gpfs_vfs_readpage**: Number of readpage operations.
- **gpfs_vfs_readpage_t**: Amount of time in seconds spent in readpage operations.
- **gpfs_vfs_remove**: Number of remove operations.
- **gpfs_vfs_remove_t**: Amount of time in seconds spent in remove operations.
- **gpfs_vfs_removexattr**: Number of removexattr operations.
- **gpfs_vfs_removexattr_t**: Amount of time in seconds spent in removexattr operations.
- **gpfs_vfs_rename**: Number of rename operations.
- **gpfs_vfs_rename_t**: Amount of time in seconds spent in rename operations.
- **gpfs_vfs_rmdir**: Number of rmdir operations.
- **gpfs_vfs_rmdir_t**: Amount of time in seconds spent in rmdir operations.
- **gpfs_vfs_setacl**: Number of setacl operations.
- **gpfs_vfs_setacl_t**: Amount of time in seconds spent in setacl operations.
- **gpfs_vfs_setattr**: Number of setattr operations.
- **gpfs_vfs_setattr_t**: Amount of time in seconds spent in setattr operations.
- **gpfs_vfs_setxattr**: Number of setxattr operations.
- **gpfs_vfs_setxattr_t**: Amount of time in seconds spent in setxattr operations.
- **gpfs_vfs_statfs**: Number of statfs operations.
- **gpfs_vfs_statfs_t**: Amount of time in seconds spent in statfs operations.
- **gpfs_vfs_symlink**: Number of symlink operations.
- **gpfs_vfs_symlink_t**: Amount of time in seconds spent in symlink operations.
- **gpfs_vfs_sync**: Number of sync operations.
- **gpfs_vfs_sync_t**: Amount of time in seconds spent in sync operations.
- **gpfs_vfs_tsfattr**: Number of tsfsattr operation.
- **gpfs_vfs_tsfattr_t**: Amount of time in seconds spent in tsfattr operations.
- **gpfs_vfs_tsfsattr**: Number of tsfattr operations.
- **gpfs_vfs_tsfsattr_t**: Amount of time in seconds spent in tsfsattr operations.
- **gpfs_vfs_unmap**: Number of unmap operations.

- **gpfs_vfs_unmap_t**: Amount of time in seconds spent in unmap operations.
- **gpfs_vfs_vget**: Number of vget operations.
- **gpfs_vfs_vget_t**: Amount of time in seconds spent in vget operations.
- **gpfs_vfs_write**: Number of write operations.
- **gpfs_vfs_write_t**: Amount of time in seconds spent in write operations.
- **gpfs_vfs_writepage**: Number of writepage operations.
- **gpfs_vfs_writepage_t**: Amount of time in seconds spent in writepage operations.

Computed Metrics

The following metrics are computed for GPFS:
- **gpfs_write_avg_lat (latency)**: gpfs_vfs_write_t / gpfs_vfs_write
- **gpfs_read_avg_lat (latency)**: gpfs_vfs_read_t / gpfs_vfs_read
- **gpfs_create_avg_lat (latency)**: gpfs_vfs_create_t / gpfs_vfs_create
- **gpfs_remove_avg_lat (latency)**: gpfs_vfs_remove_t / gpfs_vfs_remove

NFS

Native Metrics
- **nfs_read_req**: Number of bytes requested for reading.
- **nfs_write_req**: Number of bytes requested for writing.
- **nfs_read**: Number of bytes transferred for reading.
- **nfs_write**: Number of bytes transferred for writing.
- **nfs_read_ops**: Number of total read operations.
- **nfs_write_ops**: Number of total write operations.
- **nfs_read_err**: Number of erroneous read operations.
- **nfs_write_err**: Number of erroneous write operations.
- **nfs_read_lat**: Time consumed by read operations (in ns).
- **nfs_write_lat**: Time consumed by write operations (in ns).
- **nfs_read_queue**: Time spent in the rpc wait queue.
- **nfs_write_queue**: Time spent in the rpc wait queue.

Computed Metrics

The following metrics are computed for NFS:
- **nfs_total_ops**: nfs_read_ops + nfs_write_ops
- **nfsIOlatencyRead**: (nfs_read_lat + nfs_read_queue) / nfs_read_ops
- **nfsIOlatencyWrite**: (nfs_write_lat + nfs_write_queue) / nfs_write_ops
- **nfsReadOpThroughput**: nfs_read/nfs_read_ops
- **nfsWriteOpThroughput**: nfs_write/nfs_write_ops

Object

Native Metrics

ObjectAccount
- **account_auditor_time**: Timing data for individual account database audits.
- **account_reaper_time**: Timing data for each reap_account() call.

- **account_replicator_time**: Timing data for each database replication attempt not resulting in a failure.
- **account_DEL_time**: Timing data for each DELETE request not resulting in an error.
- **account_DEL_err_time**: Timing data for each DELETE request resulting in an error: bad request, not mounted, missing timestamp.
- **account_GET_time**: Timing data for each GET request not resulting in an error.
- **account_GET_err_time**: Timing data for each GET request resulting in an error: bad request, not mounted, bad delimiter, account listing limit too high, bad accept header.
- **account_HEAD_time**: Timing data for each HEAD request not resulting in an error.
- **account_HEAD_err_time**: Timing data for each HEAD request resulting in an error: bad request, not mounted.
- **account_POST_time**: Timing data for each POST request not resulting in an error.
- **account_POST_err_time**: Timing data for each POST request resulting in an error: bad request, bad or missing timestamp, not mounted.
- **account_PUT_time**: Timing data for each PUT request not resulting in an error.
- **account_PUT_err_time**: Timing data for each PUT request resulting in an error: bad request, not mounted, conflict, recently-deleted.
- **account_REPLICATE_time**: Timing data for each REPLICATE request not resulting in an error.
- **account_REPLICATE_err_time**: Timing data for each REPLICATE request resulting in an error: bad request, not mounted.

ObjectContainer

- **container_auditor_time**: Timing data for each container audit.
- **container_replicator_time**: Timing data for each database replication attempt not resulting in a failure.
- **container_DEL_time**: Timing data for each DELETE request not resulting in an error.
- **container_DEL_err_time**: Timing data for DELETE request errors: bad request, not mounted, missing timestamp, conflict.
- **container_GET_time**: Timing data for each GET request not resulting in an error.
- **container_GET_err_time**: Timing data for GET request errors: bad request, not mounted, parameters not utf8, bad accept header.
- **container_HEAD_time**: Timing data for each HEAD request not resulting in an error.
- **container_HEAD_err_time**: Timing data for HEAD request errors: bad request, not mounted.
- **container_POST_time**: Timing data for each POST request not resulting in an error.
- **container_POST_err_time**: Timing data for POST request errors: bad request, bad x-container-sync-to, not mounted.
- **container_PUT_time**: Timing data for each PUT request not resulting in an error.
- **container_PUT_err_time**: Timing data for PUT request errors: bad request, missing timestamp, not mounted, conflict.
- **container_REPLICATE_time**: Timing data for each REPLICATE request not resulting in an error.
- **container_REPLICATE_err_time**: Timing data for REPLICATE request errors: bad request, not mounted.
- **container_sync_deletes_time**: Timing data for each container database row synchronization via deletion.
- **container_sync_puts_time**: Timing data for each container database row synchronization via PUTing.
- **container_updater_time**: Timing data for processing a container; only includes timing for containers which needed to update their accounts.

ObjectObject

- **object_auditor_time**: Timing data for each object audit (does not include any rate-limiting sleep time for max_files_per_second, but does include rate-limiting sleep time for max_bytes_per_second).
- **object_expirer_time**: Timing data for each object expiration attempt, including ones resulting in an error.
- **object_replicator_partition_delete_time**: Timing data for partitions replicated to another node because they didn't belong on this node. This metric is not tracked per device.
- **object_replicator_partition_update_time**: Timing data for partitions replicated which also belong on this node. This metric is not tracked per-device.
- **object_DEL_time**: Timing data for each DELETE request not resulting in an error.
- **object_DEL_err_time**: Timing data for DELETE request errors: bad request, missing timestamp, not mounted, precondition failed. Includes requests which couldn't find or match the object.
- **object_GET_time**: Timing data for each GET request not resulting in an error. Includes requests which couldn't find the object (including disk errors resulting in file quarantine).
- **object_GET_err_time**: Timing data for GET request errors: bad request, not mounted, header timestamps before the epoch, precondition failed. File errors resulting in a quarantine are not counted here.
- **object_HEAD_time**: Timing data for each HEAD request not resulting in an error. Includes requests which couldn't find the object (including disk errors resulting in file quarantine).
- **object_HEAD_err_time**: Timing data for HEAD request errors: bad request, not mounted.
- **object_POST_time**: Timing data for each POST request not resulting in an error.
- **object_POST_err_time**: Timing data for POST request errors: bad request, missing timestamp, delete-at in past, not mounted.
- **object_PUT_time**: Timing data for each PUT request not resulting in an error.
- **object_PUT_err_time**: Timing data for PUT request errors: bad request, not mounted, missing timestamp, object creation constraint violation, delete-at in past.
- **object_REPLICATE_time**: Timing data for each REPLICATE request not resulting in an error.
- **object_REPLICATE_err_time**: Timing data for REPLICATE request errors: bad request, not mounted.
- **object_updater_time**: Timing data for object sweeps to flush async_pending container updates. Does not include object sweeps which did not find an existing async_pending storage directory.

ObjectProxy

- **proxy_account_latency**: Timing data up to completion of sending the response headers, 200: standard response for successful HTTP requests.
- **proxy_container_latency**: Timing data up to completion of sending the response headers, 200: standard response for successful HTTP requests.
- **proxy_object_latency**: Timing data up to completion of sending the response headers, 200: standard response for successful HTTP requests.
- **proxy_account_GET_time**: Timing data for GET request, start to finish, 200: standard response for successful HTTP requests
- **proxy_account_GET_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 200: standard response for successful HTTP requests.
- **proxy_account_HEAD_time**: Timing data for HEAD request, start to finish, 204: request processed, no content returned.
- **proxy_account_HEAD_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 204: request processed, no content returned.
- **proxy_container_DEL_time**: Timing data for DELETE request, start to finish, 204: request processed, no content returned.
- **proxy_container_DEL_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 204: request processed, no content returned.

- **proxy_container_GET_time**: Timing data for GET request, start to finish, 200: standard response for successful HTTP requests.
- **proxy_container_GET_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 200: standard response for successful HTTP requests.
- **proxy_container_HEAD_time**: Timing data for HEAD request, start to finish, 204: request processed, no content returned.
- **proxy_container_HEAD_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 204: request processed, no content returned. 1
- **proxy_container_PUT_time**: Timing data for each PUT request not resulting in an error, 201: request has been fulfilled; new resource created.
- **proxy_container_PUT_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 201: request has been fulfilled; new resource created.
- **proxy_object_DEL_time**: Timing data for DELETE request, start to finish, 204: request processed, no content returned.
- **proxy_object_DEL_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 204: request processed, no content returned.
- **proxy_object_GET_time**: Timing data for GET request, start to finish, 200: standard response for successful HTTP requests.
- **proxy_object_GET_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 200: standard response for successful HTTP requests.
- **proxy_object_HEAD_time**: Timing data for HEAD request, start to finish, 200: request processed, no content returned.
- **proxy_object_HEAD_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, , 200: request processed, no content returned.
- **proxy_object_PUT_time**: Timing data for each PUT request not resulting in an error, 201: request has been fulfilled; new resource created.
- **proxy_object_PUT_bytes**: The sum of bytes transferred in (from clients) and out (to clients) for requests, 201: request has been fulfilled; new resource created.

Note: There are no computed metrics for Object protocol.

SMB

SMBGlobalStats

- **connect count**: Number of connections since startup of parent smbd process
- **disconnect count**: Number of connections closed since startup
- **idle**: Describes idling behavior of smbds
  - **count**: Number of times the smbd processes are waiting for events in epoll
  - **time**: Times the smbd process spend in epoll waiting for events
- **cpu_user time**: The user time determined by the get_rusage system call in seconds
- **cpu_system time**: The system time determined by the get_rusage system call in seconds
- **request count**: Number of SMB requests since startup
- **push_sec_ctx**: Smbds switch between the user and the root security context; push allows to put the current context onto a stack
  - **count**: Number of time the current security context is pushed onto the stack
  - **time**: The time it takes to put the current security context; this includes all syscalls required to save the current context on the stack
- **pop_sec_ctx**: Getting the last security context from the stackand restore it
  - **count**: Number of times the current security context is restored from the stack

  - **time**: The time it takes to put the restore the security context from the stack; this includes all syscalls required to get restore the security context from the stack
- **set_sec_ctx**:
  - **count**: Number of times the security context is set for user
  - **time**: The time it takes to set the security context for user
- **set_root_sec_ctx**:
  - **count**: Number of times the security context is set for user
  - **time**: The time it takes to set the security context for user

SMB2 metrics

These metrics are available for all of the following areas:
- **op_count**: Number of times the corresponding SMB request has been called.
- **op_idle**
  - **for notify**: Time between notification request and a corresponding notification being sent
  - **for oplock breaks**: Time waiting until an oplock is broken
  - for all others the value is always zero
- **op_inbytes**: Number of bytes received for the corresponding request including protocol headers
- **op_outbytes**: Number of bytes sent for the corresponding request including protocol headers.
- **op_time**: The total amount of time spent for all corresponding SMB2 requests.

CTDB

- **CTDB version**: Version of the CTDB protocol used by the node.
- **Current time of statistics**: Time when the statistics are generated. This is useful when collecting statistics output periodically for post-processing.
- **Statistics collected since**: Time when CTDB was started or the last time statistics was reset. The output shows the duration and the timestamp.
- **num_clients**: Number of processes currently connected to CTDB's unix socket. This includes recovery daemon, CTDB tool and SMB processes (smbd, winbindd).
- **frozen**: 1 if the databases are currently frozen, 0 if otherwise.
- **recovering**: 1 if recovery is active, 0 if otherwise.
- **num_recoveries**: Number of recoveries since the start of CTDB or since the last statistics reset.
- **client_packets_sent**: Number of packets sent to client processes via unix domain socket.
- **client_packets_recv**: Number of packets received from client processes via unix domain socket.
- **node_packets_sent**: Number of packets sent to the other nodes in the cluster via TCP.
- **node_packets_recv**: Number of packets received from the other nodes in the cluster via TCP.
- **keepalive_packets_sent**: Number of keepalive messages sent to other nodes. CTDB periodically sends keepalive messages to other nodes. See KeepaliveInterval tunable in CTDB-tunables(7) for more details.
- **keepalive_packets_recv**: Number of keepalive messages received from other nodes.
- node: This section lists various types of messages processed which originated from other nodes via TCP.
  - **req_call**: Number of REQ_CALL messages from the other nodes.
  - **reply_call**: Number of REPLY_CALL messages from the other nodes.
  - **req_dmaster**: Number of REQ_DMASTER messages from the other nodes.
  - **reply_dmaster**: Number of REPLY_DMASTER messages from the other nodes.
  - **reply_error**: Number of REPLY_ERROR messages from the other nodes.
  - **req_message**: Number of REQ_MESSAGE messages from the other nodes.

  - **req_control**: Number of REQ_CONTROL messages from the other nodes.
  - **reply_control**: Number of REPLY_CONTROL messages from the other nodes.
- **client**: This section lists various types of messages processed which originated from clients via unix domain socket.
  - **req_call**: Number of REQ_CALL messages from the clients.
  - **req_message**: Number of REQ_MESSAGE messages from the clients.
  - **req_control**: Number of REQ_CONTROL messages from the clients.
- **timeouts**: This section lists timeouts occurred when sending various messages.
  - **call**: Number of timeouts for REQ_CALL messages.
  - **control**: Number of timeouts for REQ_CONTROL messages.
  - **traverse**: Number of timeouts for database traverse operations.
- **locks**: This section lists locking statistics.
  - **num_calls**: Number of completed lock calls. This includes database locks and record locks.
  - **num_current**: Number of scheduled lock calls. This includes database locks and record locks.
  - **num_pending**: Number of queued lock calls. This includes database locks and record locks.
  - **num_failed**: Number of failed lock calls. This includes database locks and record locks.
- **total_calls**: Number of req_call messages processed from clients. This number should be same as client --> req_call.
- **pending_calls**: Number of req_call messages which are currently being processed. This number indicates the number of record migrations in flight.
- **childwrite_calls**: Number of record update calls. Record update calls are used to update a record under a transaction.
- **pending_childwrite_calls**: Number of record update calls currently active.
- **memory_used**: The amount of memory in bytes currently used by CTDB using talloc. This includes all the memory used for CTDBÂ´s internal data structures. This does not include the memory mapped TDB databases.
- **max_hop_count**: The maximum number of hops required for a record migration request to obtain the record. High numbers indicate record contention.
- **total_ro_delegations**: Number of readonly delegations created.
- **total_ro_revokes**: Number of readonly delegations that were revoked. The difference between total_ro_revokes and total_ro_delegations gives the number of currently active readonly delegations.
- **hop_count_buckets**: Distribution of migration requests based on hop counts values.
- **lock_buckets**: Distribution of record lock requests based on time required to obtain locks. Buckets are < 1ms, < 10ms, < 100ms, < 1s, < 2s, < 4s, < 8s, < 16s, < 32s, < 64s, > 64s.
- **locks_latency**: The minimum, the average and the maximum time (in seconds) required to obtain record locks.
- **reclock_ctdbd**: The minimum, the average and the maximum time (in seconds) required to check if recovery lock is still held by recovery daemon when recovery mode is changed. This check is done in ctdb daemon.
- **reclock_recd**: The minimum, the average and the maximum time (in seconds) required to check if recovery lock is still held by recovery daemon during recovery. This check is done in recovery daemon.
- **call_latency**: The minimum, the average and the maximum time (in seconds) required to process a REQ_CALL message from client. This includes the time required to migrate a record from remote node, if the record is not available on the local node.
- **childwrite_latency**: The minimum, the average and the maximum time (in seconds) required to update records under a transaction.

Cross Protocol

- **nfs_iorate_read_perc**: nfs_read_ops/(op_count+nfs_read_ops)
- **nfs_iorate_read_perc_exports**: 1.0*nfs_read_ops/(op_count+nfs_read_ops)
- **nfs_iorate_write_perc**: nfs_write_ops/(write|op_count+nfs_write_ops)
- **nfs_iorate_write_perc_exports**: 1.0*nfs_write_ops/(op_count+nfs_write_ops)
- **nfs_read_throughput_perc**: nfs_read/(read|op_outbytes+nfs_read)
- **nfs_write_throughput_perc**: nfs_write/(write|op_outbytes+nfs_write)
- **smb_iorate_read_perc**: op_count/(op_count+nfs_read_ops)
- **smb_iorate_write_perc**: op_count/(op_count+nfs_write_ops)
- **smb_latency_read**: read|op_time/read|op_count
- **smb_latency_write**: write|op_time/write|op_count
- **smb_read_throughput_perc**: read|op_outbytes/(read|op_outbytes+nfs_read)
- **smb_total_cnt**: write|op_count+close|op_count
- **smb_tp**: op_inbytes+op_outbytes
- **smb_write_throughput_perc**: write|op_outbytes/(write|op_outbytes+nfs_write)
- **total_read_throughput**: nfs_read+read|op_outbytes
- **total_write_throughput**: nfs_write+write|op_inbytes

Performance monitoring through IBM Spectrum Scale GUI

The IBM Spectrum Scale GUI provides a graphical representation of the status and historical trends of the key performance indicators. This helps the users to make decisions easily without wasting time. Use **Monitoring** > **Dashboard** and **Monitoring** > **Performance** pages in the GUI to monitor the performance of the system based on various aspects.

Dashboards

The dashboard is an easy to read, single page, and real-time user interface that provides a quick overview of the system performance.

The dashboard consists of several dashboard widgets and the associated favorite charts that can be displayed within a chosen layout. There are various dashboard widgets available and the number of widgets is expected to grow in the following releases of this product. Currently, the following important widgets are available in the dashboard:

- Performance
- File system capacity by fileset
- System health events
- System overview
- Filesets with the largest growth rate in last week
- Timeline

To select the widgets that need to be displayed in the dashboard, click the ellipsis icon (**...**) at the upper right corner of the page and select **Edit Widgets**. Now, you can edit the existing widgets on the dashboard by using the edit option available with the individual widgets and add new widgets in the dashboard by using the **Add Widget** option.

Favorite charts

Favorite charts and predefined charts are available for selection when you add widgets in the dashboard. Favorite charts are customized predefined charts.

-

  To display favorite charts, click the Down Arrow icon in the **Search** box.
- To create a favorite chart, in the Performance page, click the 'star' icon next to the chart title. Enter the label and select single chart or dual chart mode. A favorite chart with single-chart configuration can be displayed on the Dashboard page.

Dashboard layout

The dashboard layout is stored per browser. That is, a different dashboard layout can be configured per browser. To change or select a layout for the widgets that are displayed in the dashboard, click the button on the upper right corner of the Dashboard page and select **Layout Options**. You can select a pre-defined layout or create a customized layout.

Performance

Use the Performance page to monitor the performance of system resources and file and object storage. Performance of the system can be monitored by using various pre-defined charts. You can select the required charts and monitor the performance based on the filter criteria.

The pre-defined performance charts and metrics help in investigating every node or any particular node that is collecting the metrics.

Configuring performance data collection

The **`mmperfmon`** command can be used to query performance data through CLI, and configure the performance data collection. The GUI displays a subset of the available metrics. The data that is used by the performance monitoring tool is collected from the following components:
- **Collector:** The metric collector runs on a single node and gathers metrics from all the nodes that are running the associated sensors. The metrics are stored in a database on the collector node for future review. The collector can run on any node in the system. By default, the collector runs on the administrator node, which is usually the installation node.
- **Sensor:** The sensors run on any node that is required to collect metrics. By default, the sensors are started on every protocol node. On GPFS-only nodes, the sensors are not started by default.

Note: To retrieve metrics for SMB, NFS, or Object, these protocols must be active on the specific node.

Viewing performance charts

The charting section displays the performance details based on various aspects. The GUI provides a rich set of controls to view performance charts. You can use these controls to perform the following actions on the charts that are displayed in the page:
- Zoom the chart by using the mouse wheel or resizing the timeline control. Y-axis can be automatically adjusted during zooming.
- Click and drag the chart or the timeline control at the bottom. Y-axis can be automatically adjusted during panning.
- Compare charts side by side. You can synchronize y-axis and bind x-axis in the dual chart mode. Click the 'star' symbol that is placed next to the chart title and select whether you need to use the single chart or dual chart mode. To modify the x and y axes of the chart in the dual chart mode, click the configuration symbol next to the title *Performance* and select the required options.
- Link the timelines of the two charts can be linked together by using the display options that are available.
- The Dashboard allows to access all single graph charts, which are either predefined or custom created favorites.

Modifying and creating new charts

You can edit an existing chart by clicking the icon that is available on the upper right corner of the performance chart and select Edit to modify the Metrics selections. Do the following to drill down to the metric you are interested in:

1. Select **Resource type**. This is the area from which the data is taken to create the performance analysis.
2. Select **Aggregation level**. The aggregation level determines the level at which the data is aggregated.
3. Select a time range. The time range is the interval during which the data is gathered. You can use this feature to find entities with very low or very high metrics.
4. Select the entities that need to be graphed. The table lists all entities that are available for the chosen resource type and aggregation level. When a metric is selected, you can also see the selected metrics in the same grid and use methods like sorting, filtering, or adjusting the time frame to select the entities that you want to select.
5. Select **Metrics**. Metrics is the type of data that needs to be included in the performance chart. The aggregation levels that are available for selection varies based on the resource type. The following table lists the resource types and the corresponding available aggregation levels.

The aggregation levels that are available for selection varies based on the resource type. The following table lists the resource types and the corresponding available aggregation levels.

Table 7. Resource types and corresponding aggregation levels

Resource types	**Aggregation levels**
Network	Node
	Adapters per node
	Adapter
	Cluster
System Resources	Node
	Cluster
IBM Spectrum Scale Server	NSD
	File systems
	Pool
	Cluster
IBM Spectrum Scale Client	File systems
	Cluster
	Node
NFS	Cluster
	Node
	Export
SMB	Node
	Cluster
Object	Node
	Cluster
CTDB	Node
	Database per node
	Cluster

Chapter 10. Monitoring system health through the IBM Spectrum Scale GUI

Use **Monitoring** > **Events** page to monitor the events that are reported in the system. The Events page displays events and you can monitor and troubleshoot errors on your system.

There are three options to filter events by their status:
- **Current Issues** displays all unfixed errors and warnings.
- **Unread Messages** displays all unfixed errors and warnings and information messages that are not marked as read.
- **All Events** displays every event, no matter if it is fixed or marked as read.

The status icons help to quickly determine whether the event is informational, a warning, or an error. Click an event and select **Properties** from the **Action** menu to see detailed information on the event. The event table displays the most recent events first.

Marking events as Read

You can mark certain events as read to change the status of the event in the events view. The status icons become gray in case an error or warning is fixed or if it is marked as read.

There are events on states that start with "MS*". These events can be errors, warnings, or information messages that cannot be marked as read and these events automatically changes the status from current to historic when the problem is resolved or information condition changes. The user must either fix the problem or change the state of some component to make the current event a historical event. There are also message events that start with MM*. These events never become historic by itself. The user must use the action **Mark as Read** on those events to make them historical because the system cannot detect itself even if the problem or information is not valid anymore.

Running fix procedure

Some issues can be resolved by running a fix procedure. Use action **Run Fix Procedure**to do so. The Events page provides a recommendation for which fix procedure to run next.

Setting up event notifications

The system can use Simple Network Management Protocol (SNMP) traps and emails to notify you when significant events are detected. Any combination of these notification methods can be used simultaneously. Use **Settings** > **Event Notifications** page in the GUI to configure event notifications.

Notifications are normally sent immediately after an event is raised. Each event that the system detects is assigned an event type of Alert or Messages. When you configure notifications, specify whether the notifications must be sent and which notification types need to be sent to that recipient.

The following table describes the levels of event notifications.

Table 8. Notification levels

Notification level	Description
Critical	Critical notification is sent to indicate a problem that must be corrected as soon as possible. This notification indicates a serious problem with the system. For example, the event that is being reported might indicate a loss of redundancy in the system, and it is possible that another failure might result in loss of access to data. The most typical reason that this type of notification is because of a hardware failure, but some configuration errors or fabric errors also are included in this notification level. **Note:** Hardware failures are reported only on ESS and GSS.
Warning	A warning notification is sent to indicate a problem or unexpected condition with the system. Always immediately investigate this type of notification to determine the effect that it might have on your operation, and make any necessary corrections. A warning notification does not require any replacement parts and therefore it does not require IBM Support Center involvement.
Information	An informational notification is sent to indicate that an expected event is occurred. For example, a NAS service is started. No remedial action is required when these notifications are sent.

The following table describes the event types.

Table 9. Event types

Event type	Description
Alerts	An Alert is sent to indicate a problem that you need to investigate and resolve immediately. These event types can indicate a serious problem with the system.
Messages	A message is an informational event that shows status of operations that are running on the system.

Configuring email notifications

The email feature transmits operational and error-related data in the form of an event notification email.

To configure an email server, from the Event Notifications page, select **Email Server**. Select **Edit** and then click **Enable email notifications**. Enter required details and when you are ready, click **OK**.

Emails are sent to the recipients to intimate them about the quota reports or if there are issues or events in the clustered file system.

To create email recipients, select **Email Recipients** from the **Event Notifications** page, and then click **Create Recipient**.

Note: You can change the email notification configuration or disable the email service at any time.

Configuring SNMP server

Simple Network Management Protocol (SNMP) is a standard protocol for managing networks and exchanging messages. The system can send SNMP messages that notify personnel about an event. You can use an SNMP manager to view the SNMP messages that the system sends.

With an SNMP manager, such as IBM Systems Director, you can view, and act on, the messages that the SNMP agent sends. The SNMP manager can send emails when an event occurs in the system. You can specify up to a maximum of six SNMP servers.

Chapter 11. Monitoring GPFS I/O performance with the mmpmon command

Use the **mmpmon** command to monitor GPFS performance on the node in which it is run, and other specified nodes.

Before attempting to use the **mmpmon** command, review the command documentation in the *IBM Spectrum Scale: Administration and Programming Reference*.

Next, read all of the following relevant **mmpmon** topics.
- "Overview of mmpmon"
- "Specifying input to the mmpmon command"
- "Example mmpmon scenarios and how to analyze and interpret their results" on page 186
- "Other information about mmpmon output" on page 195

Overview of mmpmon

The **mmpmon** facility allows the system administrator to collect I/O statistics from the point of view of GPFS servicing application I/O requests.

The collected data can be used for many purposes, including:
- Tracking I/O demand over longer periods of time - weeks or months.
- Recording I/O patterns over time (when peak usage occurs, and so forth).
- Determining if some nodes service more application demand than others.
- Monitoring the I/O patterns of a single application which is spread across multiple nodes.
- Recording application I/O request service times.

Figure 10 shows the software layers in a typical system with GPFS. **mmpmon** is built into GPFS.

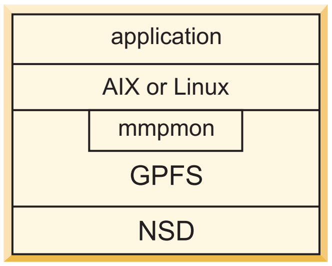


Figure 10. Node running mmpmon

Specifying input to the mmpmon command

The input requests to the **mmpmon** command allow the system administrator to collect I/O statistics per mounted file system (**fs_io_s**) or for the entire node (**io_s**).

The **mmpmon** command must be run using root authority. For command syntax, see **mmpmon** in the *IBM Spectrum Scale: Administration and Programming Reference*.

The **mmpmon** command is controlled by an input file that contains a series of requests, one per line. This input can be specified with the **-i** flag, or read from standard input (stdin). Providing input using stdin allows **mmpmon** to take keyboard input or output piped from a user script or application.

Leading blanks in the input file are ignored. A line beginning with a pound sign (#) is treated as a comment. Leading blanks in a line whose first non-blank character is a pound sign (#) are ignored.

Table 10 describes the **mmpmon** input requests.

Table 10. Input requests to the **mmpmon** *command*

Request	**Description**
fs_io_s	"Display I/O statistics per mounted file system" on page 159
io_s	"Display I/O statistics for the entire node" on page 160
nlist add *name*[*name*...]	"Add node names to a list of nodes for mmpmon processing" on page 162
nlist del	"Delete a node list" on page 164
nlist new *name*[*name*...]	"Create a new node list" on page 164
nlist s	"Show the contents of the current node list" on page 165
nlist sub *name*[*name*...]	"Delete node names from a list of nodes for mmpmon processing" on page 166
once *request*	Indicates that the request is to be performed only once.
reset	"Reset statistics to zero" on page 169
rhist nr	"Changing the request histogram facility request size and latency ranges" on page 172
rhist off	"Disabling the request histogram facility" on page 174. This is the default.
rhist on	"Enabling the request histogram facility" on page 175
rhist p	"Displaying the request histogram facility pattern" on page 176
rhist reset	"Resetting the request histogram facility data to zero" on page 179
rhist s	"Displaying the request histogram facility statistics values" on page 180
rpc_s	"Displaying the aggregation of execution time for Remote Procedure Calls (RPCs)" on page 182
rpc_s size	"Displaying the Remote Procedure Call (RPC) execution time according to the size of messages" on page 184
source *filename*	"Using request *source* and prefix directive *once*" on page 189
ver	"Displaying mmpmon version" on page 185
vio_s	"Displaying vdisk I/O statistics". See *IBM Spectrum Scale RAID: Administration* for more information.
vio_s_reset	"Resetting vdisk I/O statistics". See *IBM Spectrum Scale RAID: Administration* for more information.

Running mmpmon on multiple nodes

Invoke **mmpmon** list requests on a single node for mmpmon request processing on multiple nodes in a local cluster.

The **mmpmon** command may be invoked on one node to submit requests to multiple nodes in a local GPFS cluster by using the **nlist** requests. See "Understanding the node list facility" on page 162.

Running mmpmon concurrently from multiple users on the same node

Multiple instances of **mmpmon** can run on the same node so that different performance analysis applications and scripts can use the same performance data.

Five instances of **mmpmon** may be run on a given node concurrently. This is intended primarily to allow different user-written performance analysis applications or scripts to work with the performance data. For example, one analysis application might deal with **fs_io_s** and **io_s** data, while another one deals with **rhist** data, and another gathers data from other nodes in the cluster. The applications might be separately written or separately maintained, or have different sleep and wake-up schedules.

Be aware that there is only one set of counters for **fs_io_s** and **io_s** data, and another, separate set for **rhist** data. Multiple analysis applications dealing with the same set of data must coordinate any activities that could reset the counters, or in the case of **rhist** requests, disable the feature or modify the ranges.

Display I/O statistics per mounted file system

The **fs_io_s** input request to the **mmpmon** command allows the system administrator to collect I/O statistics per mounted file system.

The **fs_io_s** (file system I/O statistics) request returns strings containing I/O statistics taken over all mounted file systems as seen by that node, and are presented as total values for each file system. The values are cumulative since the file systems were mounted or since the last **reset** request, whichever is most recent. When a file system is unmounted, its statistics are lost.

Read and write statistics are recorded separately. The statistics for a given file system are for the file system activity on the node running **mmpmon**, not the file system in total (across the cluster).

Table 11 describes the keywords for the **fs_io_s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 11. Keywords and values for the **mmpmon fs_io_s** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_cl_	Name of the cluster that owns the file system.
_fs_	The name of the file system for which data are being presented.
_d_	The number of disks in the file system.
_br_	Total number of bytes read, from both disk and cache.
_bw_	Total number of bytes written, to both disk and cache.
_oc_	Count of **open()** call requests serviced by GPFS. This also includes **creat()** call counts.
_cc_	Number of **close()** call requests serviced by GPFS.
rdc	Number of application read requests serviced by GPFS.
_wc_	Number of application write requests serviced by GPFS.
dir	Number of **readdir()** call requests serviced by GPFS.
_iu_	Number of inode updates to disk.

Example of mmpmon fs_io_s request

This is an example of the **fs_io_s** input request to the **mmpmon** command and the resulting output that displays the I/O statistics per mounted file system.

Assume that **commandFile** contains this line:

```
fs_io_s
```

and this command is issued:

```
mmpmon -p  -i commandFile
```

The output is two lines in total, and similar to this:

```
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407431 _cl_ myCluster.xxx.com
_fs_ gpfs2 _d_ 2 _br_ 6291456 _bw_ 314572800 _oc_ 10 _cc_ 16 _rdc_ 101 _wc_ 300 _dir_ 7 _iu_ 2
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407455 _cl_ myCluster.xxx.com
_fs_ gpfs1 _d_ 3 _br_ 5431636 _bw_ 173342800 _oc_ 6 _cc_ 8 _rdc_ 54 _wc_ 156 _dir_ 3 _iu_ 6
```

The output consists of one string per mounted file system. In this example, there are two mounted file systems, **gpfs1** and **gpfs2**.

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:    myCluster.xxx.com
filesystem: gpfs2
disks: 2
timestamp: 1066660148/407431
bytes read: 6291456
bytes written: 314572800
opens: 10
closes: 16
reads: 101
writes: 300
readdir: 7
inode updates: 2

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:    myCluster.xxx.com
filesystem: gpfs1
disks: 3
timestamp: 1066660148/407455
bytes read: 5431636
bytes written: 173342800
opens: 6
closes: 8
reads: 54
writes: 156
readdir: 3
inode updates: 6
```

When no file systems are mounted, the responses are similar to:

```
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 1 _t_ 1066660148 _tu_ 407431 _cl_ - _fs_ -
```

The **_rc_** field is nonzero and the both the **_fs_** and **_cl_** fields contains a minus sign. If the **-p** flag is not specified, the results are similar to:

```
mmpmon node 199.18.1.8 name node1 fs_io_s status 1
no file systems mounted
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Display I/O statistics for the entire node

The **io_s** input request to the **mmpmon** command allows the system administrator to collect I/O statistics for the entire node.

The **io_s** (I/O statistics) request returns strings containing I/O statistics taken over all mounted file systems as seen by that node, and are presented as total values for the entire node. The values are cumulative since the file systems were mounted or since the last **reset**, whichever is most recent. When a file system is unmounted, its statistics are lost and its contribution to the total node statistics vanishes. Read and write statistics are recorded separately.

Table 12 describes the keywords for the **io_s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 12. Keywords and values for the **mmpmon io_s** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_br_	Total number of bytes read, from both disk and cache.
_bw_	Total number of bytes written, to both disk and cache.
_oc_	Count of **open()** call requests serviced by GPFS. The open count also includes **creat()** call counts.
_cc_	Number of **close()** call requests serviced by GPFS.
rdc	Number of application read requests serviced by GPFS.
_wc_	Number of application write requests serviced by GPFS.
dir	Number of **readdir()** call requests serviced by GPFS.
_iu_	Number of inode updates to disk. This includes inodes flushed to disk because of access time updates.

Example of mmpmon io_s request

This is an example of the **io_s** input request to the **mmpmon** command and the resulting output that displays the I/O statistics for the entire node.

Assume that **commandFile** contains this line:

```
io_s
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is one line in total, and similar to this:

```
_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407431 _br_ 6291456
_bw_ 314572800 _oc_ 10 _cc_ 16 _rdc_ 101 _wc_ 300 _dir_ 7 _iu_ 2
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 io_s OK
timestamp: 1066660148/407431
bytes read: 6291456
bytes written: 314572800
opens: 10
closes: 16
```

```
reads: 101
writes: 300
readdir: 7
inode updates: 2
```

Understanding the node list facility

The node list facility can be used to invoke **mmpmon** on multiple nodes and gather data from other nodes in the cluster. The following table describes the nlist requests for the **mmpmon**command.

Table 13. nlist requests for the **mmpmon** *command*

Request	**Description**
nlist add *name*[*name*...]	"Add node names to a list of nodes for mmpmon processing"
nlist del	"Delete a node list" on page 164
nlist new *name*[*name*...]	"Create a new node list" on page 164
nlist s	"Show the contents of the current node list" on page 165
nlist sub *name*[*name*...]	"Delete node names from a list of nodes for mmpmon processing" on page 166

When specifying node names, keep these points in mind:

1. A node name of '.' (dot) indicates the current node.
2. A node name of '*' (asterisk) indicates all currently connected local cluster nodes.
3. The nodes named in the node list must belong to the local cluster. Nodes in remote clusters are not supported.
4. A node list can contain nodes that are currently down. When an inactive node comes up, **mmpmon** will attempt to gather data from it.
5. If a node list contains an incorrect or unrecognized node name, all other entries in the list are processed. Suitable messages are issued for an incorrect node name.
6. When **mmpmon** gathers responses from the nodes in a node list, the full response from one node is presented before the next node. Data is not interleaved. There is no guarantee of the order of node responses.
7. The node that issues the **mmpmon** command need not appear in the node list. The case of this node serving only as a collection point for data from other nodes is a valid configuration.

Add node names to a list of nodes for mmpmon processing

The **nlist add** (node list add) request is used to add node names to a list of nodes for **mmpmon** to collect their data. The node names are separated by blanks.

Table 14 describes the keywords for the **nlist add** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 14. Keywords and values for the **mmpmon nlist add** *response*

Keyword	**Description**
_n_	IP address of the node processing the node list. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `add`.
_rc_	Indicates the status of the operation.

Table 14. Keywords and values for the **mmpmon nlist add** *response (continued)*

Keyword	Description
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_c_	The number of nodes in the user-supplied list.
_ni_	Node name input. A user-supplied node name from the offered list of names.
_nx_	Node name translation. The preferred GPFS name for the node.
nxip	Node name translated IP address. The preferred GPFS IP address for the node.
did	The number of nodes names considered valid and processed by the requests.
nlc	The number of nodes in the node list now (after all processing).

If the **nlist add** request is issued when no node list exists, it is handled as if it were an **nlist new** request.

Example of mmpmon nlist add request

This topic is an example of the **nlist add** request to add node names to a list of nodes for **mmpmon** processing and the output that displays.

A two- node cluster has nodes **node1** (199.18.1.2), a non-quorum node, and **node2** (199.18.1.5), a quorum node. A remote cluster has node **node3** (199.18.1.8). The **mmpmon** command is run on **node1**.

Assume that **commandFile** contains this line:

```
nlist add n2 199.18.1.2
```

and this command is issued:

```
mmpmon -p -i commandFile
```

Note in this example that an alias name **n2** was used for **node2**, and an IP address was used for **node1**. Notice how the values for **_ni_** and **_nx_** differ in these cases.

The output is similar to this:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _c_ 2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _ni_ n2 _nx_
node2 _nxip_ 199.18.1.5
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _ni_
199.18.1.2 _nx_ node1 _nxip_ 199.18.1.2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _did_ 2 _nlc_
2
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist add
initial status 0
name count 2
timestamp 1121955879/468858
node name n2, OK (name used: node2, IP address 199.18.1.5)
node name 199.18.1.2, OK (name used: node1, IP address 199.18.1.2)
final status 0
node names processed 2
current node list count 2
```

The requests **nlist add** and **nlist sub** behave in a similar way and use the same keyword and response format.

These requests are rejected if issued while quorum has been lost.

Delete a node list

The **nlist del** (node list delete) request deletes a node list if one exists. If no node list exists, the request succeeds and no error code is produced.

Table 15 describes the keywords for the **nlist del** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 15. Keywords and values for the **mmpmon nlist del** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `del`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Example of mmpmon nlist del request

This topic is an example of the **nlist del** request to delete a node list and the output that displays.

Assume that **commandFile** contains this line:

```
nlist del
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ del _rc_ 0 _t_ 1121956817 _tu_ 46050
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist del status OK timestamp 1121956908/396381
```

Create a new node list

The **nlist new** (node list new) request deletes the current node list if one exists, creates a new, empty node list, and then attempts to add the specified node names to the node list. The node names are separated by blanks.

Table 16 describes the keywords for the **nlist new** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 16. Keywords and values for the **mmpmon nlist new** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `new`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Show the contents of the current node list

The **nlist s** (node list show) request displays the current contents of the node list. If no node list exists, a count of zero is returned and no error is produced.

Table 17 describes the keywords for the **nlist s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 17. Keywords and values for the **mmpmon nlist s** *response*

Keyword	Description
_n_	IP address of the node processing the request. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `s`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_c_	Number of nodes in the node list.
mbr	GPFS preferred node name for the list member.
_ip_	GPFS preferred IP address for the list member.

Example of mmpmon nlist s request

This topic is an example of the **nlist s** request to show the contents of the current node list and the output that displays.

Assume that **commandFile** contains this line:

```
nlist s
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121956950 _tu_ 863292 _c_ 2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121956950 _tu_ 863292 _mbr_ node1
_ip_ 199.18.1.2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121956950 _tu_ 863292 _mbr_
node2 _ip_ 199.18.1.5
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist s
status 0
name count 2
timestamp 1121957505/165931
node name node1, IP address 199.18.1.2
node name node2, IP address 199.18.1.5
```

If there is no node list, the response looks like:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121957395 _tu_ 910440 _c_ 0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist s
status 0
name count 0
timestamp 1121957436/353352
the node list is empty
```

The **nlist s** request is rejected if issued while quorum has been lost. Only one response line is presented.

```
_failed_ _n_ 199.18.1.8 _nn_ node2 _rc_ 668 _t_ 1121957395 _tu_ 910440
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node2: failure status 668 timestamp 1121957395/910440
lost quorum
```

Delete node names from a list of nodes for mmpmon processing

The **nlist sub** (subtract a node from the node list) request removes a node from a list of node names.

These keywords and responses are similar to the **nlist add** request. The **_req_** keyword (action requested) for **nlist sub** is sub.

For more information, see the topic *Add a list of node names to a list of nodes for mmpmon processing*.

Node list examples and error handling

The **nlist** facility can be used to obtain GPFS performance data from nodes other than the one on which the **mmpmon** command is invoked. This information is useful to see the flow of GPFS I/O from one node to another, and spot potential problems.

A successful fs_io_s request propagated to two nodes

This topic is an example of a successful **fs_io_s** request to two nodes to display the I/O statistics per mounted file system and the resulting system output.

This command is issued:

```
mmpmon -p -i command_file
```

where **command_file** has this:

```
nlist new node1 node2
fs_io_s
```

The output is similar to this:

```
_fs_io_s_ _n_ 199.18.1.2 _nn_ node1 _rc_ 0 _t_ 1121974197 _tu_ 278619 _cl_
xxx.localdomain _fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
_dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.2 _nn_ node1 _rc_ 0 _t_ 1121974197 _tu_ 278619 _cl_
xxx.localdomain _fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
_dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974167 _tu_ 116443 _cl_
cl1.xxx.com _fs_ fs3 _d_ 3 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 3
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974167 _tu_ 116443 _cl_
cl1.xxx.comm _fs_ fs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974167 _tu_ 116443 _cl_
xxx.localdomain _fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
_dir_ 0 _iu_ 0
```

The responses from a propagated request are the same as they would have been if issued on each node separately.

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1121974088/463102
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.2 name node1 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs1
disks: 1
timestamp: 1121974088/463102
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs3
disks: 3
timestamp: 1121974058/321741
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 2

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs2
disks: 2
timestamp: 1121974058/321741
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1121974058/321741
bytes read: 0
bytes written: 0
opens: 0
closes: 0
```

```
reads: 0
writes: 0
readdir: 0
inode updates: 0
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Failure on a node accessed by mmpmon

This is an example of the system output for a failed request to two nodes to display the I/O statistics per mounted file system.

In this example, the same scenario described in "A successful fs_io_s request propagated to two nodes" on page 166 is run on **node2**, but with a failure on **node1** (a non-quorum node) because **node1** was shutdown:

```
_failed_ _n_ 199.18.1.5 _nn_ node2 _fn_ 199.18.1.2 _fnn_ node1 _rc_ 233
_t_ 1121974459 _tu_ 602231
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974459 _tu_ 616867 _cl_
cl1.xxx.com _fs_ fs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974459 _tu_ 616867 _cl_
cl1.xxx.com _fs_ fs3 _d_ 3 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974459 _tu_ 616867 _cl_
node1.localdomain _fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.5 name node2:
from node 199.18.1.2 from name node1: failure status 233 timestamp 1121974459/602231
node failed (or never started)
mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs2
disks: 2
timestamp: 1121974544/222514
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs3
disks: 3
timestamp: 1121974544/222514
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1121974544/222514
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0
```

Node shutdown and quorum loss

In this example, the quorum node (**node2**) is shutdown, causing quorum loss on **node1**. Running the same example on **node2**, the output is similar to:

```
_failed_ _n_ 199.18.1.2 _nn_ node1 _rc_ 668 _t_ 1121974459 _tu_ 616867
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1: failure status 668 timestamp 1121974459/616867
lost quorum
```

In this scenario there can be a window where **node2** is down and **node1** has not yet lost quorum. When quorum loss occurs, the **mmpmon** command does not attempt to communicate with any nodes in the node list. The goal with failure handling is to accurately maintain the node list across node failures, so that when nodes come back up they again contribute to the aggregated responses.

Node list failure values

Table 18 describes the keywords and values produced by the **mmpmon** command on a node list failure:

Table 18. Keywords and values for the **mmpmon nlist** *failures*

Keyword	Description
_n_	IP address of the node processing the node list. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_fn_	IP address of the node that is no longer responding to **mmpmon** requests.
fnn	The name by which GPFS knows the node that is no longer responding to **mmpmon** requests
_rc_	Indicates the status of the operation. See "Return codes from mmpmon" on page 196.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Reset statistics to zero

The **reset** request resets the statistics that are displayed with **fs_io_s** and **io_s** requests. The **reset** request *does not* reset the histogram data, which is controlled and displayed with **rhist** requests.

Table 19 describes the keywords for the **reset** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag. The response is a single string.

Table 19. Keywords and values for the **mmpmon reset** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Example of mmpmon reset request

This topic is an example of how to reset file system I/O and I/O statistics to zero.

Assume that **commandFile** contains this line:

```
reset
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_reset_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407431
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 reset OK
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Understanding the request histogram facility

Use the **mmpmon rhist** requests to control the request histogram facility.

The request histogram facility tallies I/O operations using a set of counters. Counters for reads and writes are kept separately. They are categorized according to a pattern that may be customized by the user. A default pattern is also provided. The **size range** and **latency range** input parameters to the **rhist nr** request are used to define the pattern.

The first time that you run the **rhist** requests, assess if there is a noticeable performance degradation. Collecting histogram data may cause performance degradation. This is possible once the histogram facility is enabled, but will probably not be noticed while the commands themselves are running. It is more of a long term issue as the GPFS daemon runs with histograms enabled.

The histogram lock is used to prevent two **rhist** requests from being processed simultaneously. If an **rhist** request fails with an **_rc_** of 16, the lock is in use. Reissue the request.

The histogram data survives file system mounts and unmounts. In order to reset this data, use the **rhist reset** request.

Table 20 describes the **rhist** requests:

Table 20. rhist requests for the **mmpmon** *command*

Request	**Description**
rhist nr	"Changing the request histogram facility request size and latency ranges" on page 172
rhist off	"Disabling the request histogram facility" on page 174. This is the default.
rhist on	"Enabling the request histogram facility" on page 175
rhist p	"Displaying the request histogram facility pattern" on page 176
rhist reset	"Resetting the request histogram facility data to zero" on page 179
rhist s	"Displaying the request histogram facility statistics values" on page 180

Specifying the size ranges for I/O histograms

The I/O histogram size ranges are used to categorize the I/O according to the size, in bytes, of the I/O operation.

The size ranges are specified using a string of positive integers separated by semicolons (;). No white space is allowed within the size range operand. Each number represents the upper bound, in bytes, of the I/O request size for that range. The numbers must be monotonically increasing. Each number may be optionally followed by the letters K or k to denote multiplication by 1024, or by the letters M or m to denote multiplication by 1048576 (1024*1024).

For example, the size range operand:

```
512;1m;4m
```

represents these four size ranges

```
0       to     512 bytes
513     to 1048576 bytes
1048577  to 4194304 bytes
4194305 and greater bytes
```

In this example, a read of size 3 MB would fall in the third size range, a write of size 20 MB would fall in the fourth size range.

A size range operand of = (equal sign) indicates that the current size range is not to be changed. A size range operand of * (asterisk) indicates that the current size range is to be changed to the default size range. A maximum of 15 numbers may be specified, which produces 16 total size ranges.

The default request size ranges are:

```
0        to      255 bytes
256      to      511 bytes
512      to     1023 bytes
1024     to     2047 bytes
2048     to     4095 bytes
4096     to     8191 bytes
8192     to    16383 bytes
16384    to    32767 bytes
32768    to    65535 bytes
65536    to   131071 bytes
131072   to   262143 bytes
262144   to   524287 bytes
524288   to  1048575 bytes
1048576  to  2097151 bytes
2097152  to  4194303 bytes
4194304  and greater bytes
```

The last size range collects all request sizes greater than or equal to 4 MB. The request size ranges can be changed by using the **rhist nr** request.

For more information, see *Processing of rhist nr*.

Specifying the latency ranges for I/O

The I/O histogram latency ranges are used to categorize the I/O according to the latency time, in milliseconds, of the I/O operation.

A full set of latency ranges are produced for each size range. The latency ranges are the same for each size range.

The latency ranges are changed using a string of positive decimal numbers separated by semicolons (;). No white space is allowed within the latency range operand. Each number represents the upper bound of the I/O latency time (in milliseconds) for that range. The numbers must be monotonically increasing. If decimal places are present, they are truncated to tenths.

For example, the latency range operand:

```
1.3;4.59;10
```

represents these four latency ranges:

```
 0.0   to   1.3   milliseconds
 1.4   to   4.5   milliseconds
 4.6   to  10.0   milliseconds
10.1  and greater milliseconds
```

In this example, a read that completes in 0.85 milliseconds falls into the first latency range. A write that completes in 4.56 milliseconds falls into the second latency range, due to the truncation.

A latency range operand of = (equal sign) indicates that the current latency range is not to be changed. A latency range operand of * (asterisk) indicates that the current latency range is to be changed to the default latency range. If the latency range operand is missing, * (asterisk) is assumed. A maximum of 15 numbers may be specified, which produces 16 total latency ranges.

The latency times are in milliseconds. The default latency ranges are:

```
0.0     to    1.0   milliseconds
1.1     to   10.0   milliseconds
10.1    to   30.0   milliseconds
30.1    to  100.0   milliseconds
100.1   to  200.0   milliseconds
200.1   to  400.0   milliseconds
400.1   to  800.0   milliseconds
800.1   to 1000.0   milliseconds
1000.1 and greater  milliseconds
```

The last latency range collects all latencies greater than or equal to 1000.1 milliseconds. The latency ranges can be changed by using the **rhist nr** request.

For more information, see *Processing of rhist nr*.

Changing the request histogram facility request size and latency ranges

The **rhist nr** (new range) request allows the user to change the size and latency ranges used in the request histogram facility.

The use of **rhist nr** implies an **rhist reset**. Counters for read and write operations are recorded separately. If there are no mounted file systems at the time **rhist nr** is issued, the request still runs. The size range operand appears first, followed by a blank, and then the latency range operand.

Table 21 describes the keywords for the **rhist nr** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 21. Keywords and values for the **mmpmon rhist nr** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `nr`.

Table 21. Keywords and values for the **mmpmon rhist nr** *response (continued)*

Keyword	Description
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Processing of rhist nr

The **rhist nr** request changes the request histogram facility request size and latency ranges.

Processing of **rhist nr** is as follows:

1. The size range and latency range operands are parsed and checked for validity. If they are not valid, an error is returned and processing terminates.
2. The histogram facility is disabled.
3. The new ranges are created, by defining the following histogram counters:
   a. Two sets, one for read and one for write.
   b. Within each set, one category for each size range.
   c. Within each size range category, one counter for each latency range.

      For example, if the user specifies 11 numbers for the size range operand and 2 numbers for the latency range operand, this produces 12 size ranges, each having 3 latency ranges, because there is one additional range for the top endpoint. The total number of counters is 72: 36 read counters and 36 write counters.
4. The new ranges are made current.
5. The old ranges are discarded. Any accumulated histogram data is lost.

The histogram facility must be explicitly enabled again using **rhist on** to begin collecting histogram data using the new ranges.

The **mmpmon** command does not have the ability to collect data only for read operations, or only for write operations. The **mmpmon** command does not have the ability to specify size or latency ranges that have different values for read and write operations. The **mmpmon** command does not have the ability to specify latency ranges that are unique to a given size range.

For more information, see *Specifying the size ranges for I/O histograms* and *Specifying the latency ranges for I/O*.

Example of mmpmon rhist nr request

This topic is an example of using **rhist nr** to change the request histogram facility request size and latency changes.

Assume that **commandFile** contains this line:

```
rhist nr 512;1m;4m 1.3;4.5;10
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ nr 512;1m;4m 1.3;4.5;10 _rc_ 0 _t_ 1078929833 _tu_ 765083
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist nr 512;1m;4m 1.3;4.5;10 OK
```

In this case, **mmpmon** has been instructed to keep a total of 32 counters. There are 16 for read and 16 for write. For the reads, there are four size ranges, each of which has four latency ranges. The same is true for the writes. They are as follows:

```
size range           0   to   512    bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
size range          513   to  1048576 bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
size range      1048577  to  4194304 bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
size range      4194305 and greater bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
```

In this example, a read of size 15 MB that completes in 17.8 milliseconds would fall in the last latency range listed here. When this read completes, the counter for the last latency range will be increased by one.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

An example of an unsuccessful response is:

```
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ nr 512;1m;4m 1;4;8;2 _rc_ 22 _t_ 1078929596 _tu_ 161683
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist nr 512;1m;4m 1;4;8;2 status 22 range error
```

In this case, the last value in the latency range, 2, is out of numerical order.

Note that the request **rhist nr = =** does not make any changes. It is ignored.

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Disabling the request histogram facility

The **rhist off** request disables the request histogram facility. This is the default value.

The data objects remain persistent, and the data they contain is not disturbed. This data is not updated again until **rhist on** is issued. **rhist off** may be combined with **rhist on** as often as desired. If there are no mounted file systems at the time **rhist off** is issued, the facility is still disabled. The response is a single string.

Table 22 on page 175 describes the keywords for the **rhist off** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 22. Keywords and values for the **mmpmon rhist off** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `off`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Example of mmpmon rhist off request

This topic is an example of the **rhist off** request to disable the histogram facility and the output that displays.

Assume that **commandFile** contains this line:

```
rhist off
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ off _rc_ 0 _t_ 1066938820 _tu_ 5755
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist off OK
```

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

```
mmpmon node 199.18.1.8 name node1 rhist off status 16
lock is busy
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Enabling the request histogram facility

The **rhist on** request enables the request histogram facility.

When **rhist on** is invoked the first time, this request creates the necessary data objects to support histogram data gathering. This request may be combined with **rhist off** (or another **rhist on**) as often as desired. If there are no mounted file systems at the time **rhist on** is issued, the facility is still enabled. The response is a single string.

Table 23 describes the keywords for the **rhist on** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 23. Keywords and values for the **mmpmon rhist on** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `on`.

Table 23. Keywords and values for the **mmpmon rhist on** *response (continued)*

Keyword	Description
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Example of mmpmon rhist on request

This topic is an example of the **rhist on** request to enable the request histogram facility and the output that displays.

Assume that **commandFile** contains this line:

```
rhist on
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ on _rc_ 0 _t_ 1066936484 _tu_ 179346
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist on OK
```

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

```
mmpmon node 199.18.1.8 name node1 rhist on status 16
lock is busy
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Displaying the request histogram facility pattern

The **rhist p** request displays the request histogram facility pattern.

The **rhist p** request returns the entire enumeration of the request size and latency ranges. The facility must be enabled for a pattern to be returned. If there are no mounted file systems at the time this request is issued, the request still runs and returns data. The pattern is displayed for both read and write.

Table 24 describes the keywords for the **rhist p** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 24. Keywords and values for the **mmpmon rhist p** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `p`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_k_	The kind, **r** or **w**, (read or write) depending on what the statistics are for.

Table 24. Keywords and values for the **mmpmon rhist p** *response (continued)*

Keyword	**Description**
_R_	Request size range, minimum and maximum number of bytes.
_L_	Latency range, minimum and maximum, in milliseconds.

The request size ranges are in bytes. The zero value used for the upper limit of the last size range means 'and above'. The request size ranges can be changed by using the **rhist nr** request.

The latency times are in milliseconds The zero value used for the upper limit of the last latency range means 'and above'. The latency ranges can be changed by using the **rhist nr** request.

The **rhist p** request allows an application to query for the entire latency pattern. The application can then configure itself accordingly. Since latency statistics are reported only for ranges with nonzero counts, the statistics responses may be sparse. By querying for the pattern, an application can be certain to learn the complete histogram set. The user may have changed the pattern using the **rhist nr** request. For this reason, an application should query for the pattern and analyze it before requesting statistics.

If the facility has never been enabled, the **_rc_** field will be nonzero. An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

If the facility has been previously enabled, the **rhist p** request will still display the pattern even if **rhist off** is currently in effect.

If there are no mounted file systems at the time **rhist p** is issued, the pattern is still displayed.

Example of mmpmon rhist p request

This topic is an example of the **rhist p** request to display the request histogram facility pattern and the output that displays.

Assume that **commandFile** contains this line:

```
rhist p
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The response contains all the latency ranges inside each of the request ranges. The data are separate for read and write:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ r
... data for reads ...
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ w
... data for writes ...
_end_
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist p OK read
... data for reads ...
mmpmon node 199.188.1.8 name node1 rhist p OK write
... data for writes ...
```

Here is an example of data for reads:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ r
_R_         0           255
_L_       0.0           1.0
_L_       1.1          10.0
_L_      10.1          30.0
_L_      30.1         100.0
_L_     100.1         200.0
_L_     200.1         400.0
_L_     400.1         800.0
_L_     800.1        1000.0
_L_    1000.1             0
_R_       256           511
_L_       0.0           1.0
_L_       1.1          10.0
_L_      10.1          30.0
_L_      30.1         100.0
_L_     100.1         200.0
_L_     200.1         400.0
_L_     400.1         800.0
_L_     800.1        1000.0
_L_    1000.1             0
_R_       512          1023
_L_       0.0           1.0
_L_       1.1          10.0
_L_      10.1          30.0
_L_      30.1         100.0
_L_     100.1         200.0
_L_     200.1         400.0
_L_     400.1         800.0
_L_     800.1        1000.0
_L_    1000.1             0
...
_R_   4194304             0
_L_       0.0           1.0
_L_       1.1          10.0
_L_      10.1          30.0
_L_      30.1         100.0
_L_     100.1         200.0
_L_     200.1         400.0
_L_     400.1         800.0
_L_     800.1        1000.0
_L_    1000.1             0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist p OK read
size range         0 to         255
  latency range    0.0 to         1.0
  latency range    1.1 to        10.0
  latency range   10.1 to        30.0
  latency range   30.1 to       100.0
  latency range  100.1 to       200.0
  latency range  200.1 to       400.0
  latency range  400.1 to       800.0
  latency range  800.1 to      1000.0
  latency range 1000.1 to           0
size range       256 to         511
  latency range    0.0 to         1.0
  latency range    1.1 to        10.0
  latency range   10.1 to        30.0
  latency range   30.1 to       100.0
  latency range  100.1 to       200.0
  latency range  200.1 to       400.0
  latency range  400.1 to       800.0
  latency range  800.1 to      1000.0
  latency range 1000.1 to           0
size range       512 to        1023
  latency range    0.0 to         1.0
  latency range    1.1 to        10.0
  latency range   10.1 to        30.0
  latency range   30.1 to       100.0
  latency range  100.1 to       200.0
  latency range  200.1 to       400.0
  latency range  400.1 to       800.0
  latency range  800.1 to      1000.0
  latency range 1000.1 to           0
...
size range   4194304 to           0
  latency range    0.0 to         1.0
  latency range    1.1 to        10.0
  latency range   10.1 to        30.0
  latency range   30.1 to       100.0
  latency range  100.1 to       200.0
  latency range  200.1 to       400.0
  latency range  400.1 to       800.0
  latency range  800.1 to      1000.0
  latency range 1000.1 to           0
```

If the facility has never been enabled, the **_rc_** field will be nonzero.

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 1 _t_ 1066939007 _tu_ 386241
```

If the **-p** flag is not specified, the output is similar to this:

```
mmpmon node 199.18.1.8 name node1 rhist p status 1
not yet enabled
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Resetting the request histogram facility data to zero

The **rhist reset** request resets the histogram statistics.

Table 25 describes the keywords for the **rhist reset** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag. The response is a single string.

Table 25. Keywords and values for the **mmpmon rhist reset** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `reset`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

If the facility has been previously enabled, the reset request will still reset the statistics even if **rhist off** is currently in effect. If there are no mounted file systems at the time **rhist reset** is issued, the statistics are still reset.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Example of mmpmon rhist reset request

This topic is an example of the **rhist reset** request to reset the histogram facility data to zero and the output that displays.

Assume that **commandFile** contains this line:

```
rhist reset
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 0 _t_ 1066939007 _tu_ 386241
```

If the **-p** flag is not specified, the output is similar to:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 0 _t_ 1066939007 _tu_ 386241
```

If the facility has never been enabled, the **_rc_** value will be nonzero:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 1 _t_ 1066939143 _tu_ 148443
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist reset status 1
not yet enabled
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Displaying the request histogram facility statistics values

The **rhist s** request returns the current values for all latency ranges which have a nonzero count.

Table 26 describes the keywords for the **rhist s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 26. Keywords and values for the **mmpmon rhist s** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `s`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_k_	The kind, **r** or **w**, (read or write) depending on what the statistics are for.
_R_	Request size range, minimum and maximum number of bytes.
_NR_	Number of requests that fell in this size range.
_L_	Latency range, minimum and maximum, in milliseconds.
_NL_	Number of requests that fell in this latency range. The sum of all **_NL_** values for a request size range equals the **_NR_** value for that size range.

If the facility has been previously enabled, the **rhist s** request will still display the statistics even if **rhist off** is currently in effect. This allows turning the histogram statistics on and off between known points and reading them later. If there are no mounted file systems at the time **rhist s** is issued, the statistics are still displayed.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Example of mmpmon rhist s request

This topic is an example of the **rhist s** request to display the request histogram facility statistics values and the output that displays.

Assume that **commandFile** contains this line:

```
rhist s
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ s _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ r
_R_      65536      131071 _NR_      32640
_L_        0.0         1.0 _NL_      25684
_L_        1.1        10.0 _NL_       4826
_L_       10.1        30.0 _NL_       1666
_L_       30.1       100.0 _NL_        464
_R_     262144      524287 _NR_       8160
_L_        0.0         1.0 _NL_       5218
_L_        1.1        10.0 _NL_        871
_L_       10.1        30.0 _NL_       1863
_L_       30.1       100.0 _NL_        208
_R_    1048576     2097151 _NR_       2040
_L_        1.1        10.0 _NL_        558
_L_       10.1        30.0 _NL_        809
_L_       30.1       100.0 _NL_        673
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ s _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ w
_R_     131072      262143 _NR_      12240
_L_        0.0         1.0 _NL_      10022
_L_        1.1        10.0 _NL_       1227
_L_       10.1        30.0 _NL_        783
_L_       30.1       100.0 _NL_        208
_R_     262144      524287 _NR_       6120
_L_        0.0         1.0 _NL_       4419
_L_        1.1        10.0 _NL_        791
_L_       10.1        30.0 _NL_        733
_L_       30.1       100.0 _NL_        177
_R_     524288     1048575 _NR_       3060
_L_        0.0         1.0 _NL_       1589
_L_        1.1        10.0 _NL_        581
_L_       10.1        30.0 _NL_        664
_L_       30.1       100.0 _NL_        226
_R_    2097152     4194303 _NR_        762
_L_        1.1         2.0 _NL_        203
_L_       10.1        30.0 _NL_        393
_L_       30.1       100.0 _NL_        166
_end_
```

This small example shows that the reports for read and write may not present the same number of ranges or even the same ranges. Only those ranges with nonzero counters are represented in the response. This is true for both the request size ranges and the latency ranges within each request size range.

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.2.5 name node1 rhist s OK timestamp 1066933849/93804 read
size range           65536 to    131071 count      32640
  latency range        0.0 to       1.0 count      25684
  latency range        1.1 to      10.0 count       4826
  latency range       10.1 to      30.0 count       1666
  latency range       30.1 to     100.0 count        464
size range          262144 to    524287 count       8160
  latency range        0.0 to       1.0 count       5218
  latency range        1.1 to      10.0 count        871
  latency range       10.1 to      30.0 count       1863
  latency range       30.1 to     100.0 count        208
size range         1048576 to   2097151 count       2040
  latency range        1.1 to      10.0 count        558
  latency range        10.1 to       30.0 count       809
  latency range        30.1 to      100.0 count       673
mmpmon node 199.18.2.5 name node1 rhist s OK timestamp 1066933849/93968 write
size range          131072 to     262143 count     12240
  latency range         0.0 to        1.0 count     10022
  latency range         1.1 to       10.0 count      1227
  latency range        10.1 to       30.0 count       783
  latency range        30.1 to      100.0 count       208
size range          262144 to     524287 count      6120
  latency range         0.0 to        1.0 count      4419
  latency range         1.1 to       10.0 count       791
  latency range        10.1 to       30.0 count       733
  latency range        30.1 to      100.0 count       177
size range          524288 to    1048575 count      3060
  latency range         0.0 to        1.0 count      1589
  latency range         1.1 to       10.0 count       581
  latency range        10.1 to       30.0 count       664
  latency range        30.1 to      100.0 count       226
size range         2097152 to     4194303 count      762
  latency range         1.1 to        2.0 count      203
  latency range        10.1 to       30.0 count      393
  latency range        30.1 to      100.0 count     166
```

If the facility has never been enabled, the **_rc_** value will be nonzero:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 1 _t_ 1066939143 _tu_ 148443
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist reset status 1
not yet enabled
```

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Understanding the Remote Procedure Call (RPC) facility

The **mmpmon** requests that start with **rpc_s** display an aggregation of execution time taken by RPCs for a time unit, for example the last 10 seconds. The statistics displayed are the average, minimum, and maximum of RPC execution time over the last 60 seconds, 60 minutes, 24 hours, and 30 days.

Table 27 describes the **rpc_s** requests:

Table 27. rpc_s requests for the **mmpmon** *command*

Request	Description
rpc_s	"Displaying the aggregation of execution time for Remote Procedure Calls (RPCs)"
rpc_s size	"Displaying the Remote Procedure Call (RPC) execution time according to the size of messages" on page 184

The information displayed with **rpc_s** is similar to what is displayed with the **mmdiag --rpc** command.

Displaying the aggregation of execution time for Remote Procedure Calls (RPCs)

The **rpc_s** request returns the aggregation of execution time for RPCs.

Table 28 on page 183 describes the keywords for the **rpc_s** response, in the order that they appear in the output.

Table 28. Keywords and values for the **mmpmon rpc_s** *response*

Keyword	Description
req	Indicates the action requested. The action can be either **size**, **node**, or **message**. If no action is requested, the default is the **rpc_s** action.
_n_	Indicates the IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	Indicates the hostname that corresponds to the IP address (the **_n_** value).
_rn_	Indicates the IP address of the remote node responding. This is the address by which GPFS knows the node. The statistics displayed are the averages from **_nn_** to this **_rnn_**.
rnn	Indicates the hostname that corresponds to the remote node IP address (the **_rn_** value). The statistics displayed are the averages from **_nn_** to this **_rnn_**.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Indicates the microseconds part of the current time of day.
rpcObj	Indicates the beginning of the statistics for **_obj_**.
obj	Indicates the RPC object being displayed.
nsecs	Indicates the number of one-second intervals maintained.
nmins	Indicates the number of one-minute intervals maintained.
nhours	Indicates the number of one-hour intervals maintained.
ndays	Indicates the number of one-day intervals maintained.
stats	Indicates the beginning of the RPC statistics.
tmu	Indicates the time unit (seconds, minutes, hours, or days).
_av_	Indicates the average value of execution time for **_cnt_** RPCs during this time unit.
min	Indicates the minimum value of execution time for **_cnt_** RPCs during this time unit.
max	Indicates the maximum value of execution time for **_cnt_** RPCs during this time unit.
cnt	Indicates the count of RPCs that occurred during this time unit.

The values allowed for **_rpcObj_** are the following:
- **AG_STAT_CHANNEL_WAIT**
- **AG_STAT_SEND_TIME_TCP**
- **AG_STAT_SEND_TIME_VERBS**
- **AG_STAT_RECEIVE_TIME_TCP**
- **AG_STAT_RPC_LATENCY_TCP**
- **AG_STAT_RPC_LATENCY_VERBS**
- **AG_STAT_RPC_LATENCY_MIXED**
- **AG_STAT_LAST**

Example of mmpmon rpc_s request

This topic is an example of the **rpc_s** request to display the aggregation of execution time for remote procedure calls (RPCs).

Assume that the file **commandFile** contains the following line:

```
rpc_s
```

The following command is issued:

```
mmpmon -p  -i commandFile
```

The output is similar to the following example:

```
_response_ begin mmpmon rpc_s
_mmpmon::rpc_s_  _req_ node _n_ 192.168.56.168 _nn_ node3 _rn_ 192.168.56.167 _rnn_ node2 _rc_ 0 _t_ 1388417709  _tu_ 641530
_rpcObj_ _obj_ AG_STAT_CHANNEL_WAIT _nsecs_ 60 _nmins_ 60 _nhours_ 24 _ndays_ 30
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
..............
..............
..............
_rpcObj_ _obj_ AG_STAT_SEND_TIME_TCP _nsecs_ 60 _nmins_ 60 _nhours_ 24 _ndays_ 30
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
............................
............................
............................
_response_ end
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Displaying the Remote Procedure Call (RPC) execution time according to the size of messages

The **rpc_s size** request returns the cached RPC-related size statistics.

Table 29 describes the keywords for the **rpc_s size** response, in the order that they appear in the output.

Table 29. Keywords and values for the **mmpmon rpc_s size** *response*

Keyword	Description
req	Indicates the action requested. In this case, the value is **rpc_s size**.
_n_	Indicates the IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	Indicates the hostname that corresponds to the IP address (the **_n_** value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Indicates the microseconds part of the current time of day.
rpcSize	Indicates the beginning of the statistics for this **_size_** group.
size	Indicates the size of the messages for which statistics are collected.
nsecs	Indicates the number of one-second intervals maintained.
nmins	Indicates the number of one-minute intervals maintained.
nhours	Indicates the number of one-hour intervals maintained.
ndays	Indicates the number of one-day intervals maintained.
stats	Indicates the beginning of the RPC-size statistics.
tmu	Indicates the time unit.
_av_	Indicates the average value of execution time for **_cnt_** RPCs during this time unit.
min	Indicates the minimum value of execution time for **_cnt_** RPCs during this time unit.
max	Indicates the maximum value of execution time for **_cnt_** RPCs during this time unit.
cnt	Indicates the count of RPCs that occurred during this time unit.

Example of mmpmon rpc_s size request

This topic is an example of the **rpc_s size** request to display the RPC execution time according to the size of messages.

Assume that the file `commandFile` contains the following line:

```
rpc_s size
```

The following command is issued:

```
mmpmon -p  -i commandFile
```

The output is similar to the following example:

```
_mmpmon::rpc_s_ _req_ size _n_ 192.168.56.167 _nn_ node2 _rc_ 0 _t_ 1388417852  _tu_ 572950
_rpcSize_ _size_ 64 _nsecs_ 60 _nmins_ 60 _nhours_ 24 _ndays_ 30
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
......................
......................
......................
_rpcSize_ _size_ 256 _nsecs_ 60 _nmins_ 60 _nhours_ 24 _ndays_ 30
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ sec _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
.....................
.....................
_stats_ _tmu_ min _av_ 0.692, _min_ 0.692, _max_ 0.692, _cnt_ 1
_stats_ _tmu_ min _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ min _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_stats_ _tmu_ min _av_ 0.000, _min_ 0.000, _max_ 0.000, _cnt_ 0
_response_ end
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Displaying mmpmon version

The **ver** request returns a string containing version information.

Table 30 Describes the keywords for the **ver** (version) response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 30. Keywords and values for the **mmpmon ver** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_v_	The version of **mmpmon**.
_lv_	The level of **mmpmon**.
_vt_	The fix level variant of **mmpmon**.

Example of mmpmon ver request

This topic is an example of the **ver** request to display the **mmpmom** version and the output that displays.

Assume that **commandFile** contains this line:

```
ver
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_ver_ _n_ 199.18.1.8 _nn_ node1 _v_ 3 _lv_ 3 _vt_ 0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 version 3.3.0
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Example mmpmon scenarios and how to analyze and interpret their results

This topic is an illustration of how **mmpmon** is used to analyze I/O data and draw conclusions based on it.

The **fs_io_s** and **io_s** requests are used to determine a number of GPFS I/O parameters and their implication for overall performance. The **rhist** requests are used to produce histogram data about I/O sizes and latency times for I/O requests. The request *source* and prefix directive *once* allow the user of **mmpmon** to more finely tune its operation.

fs_io_s and io_s output - how to aggregate and analyze the results

The **fs_io_s** and **io_s** requests can be used to determine a number of GPFS I/O parameters and their implication for overall performance.

The output from the **fs_io_s** and **io_s** requests can be used to determine:

1. The I/O service rate of a node, from the application point of view. The **io_s** request presents this as a sum for the entire node, while **fs_io_s** presents the data per file system. A rate can be approximated by taking the **_br_** (bytes read) or **_bw_** (bytes written) values from two successive invocations of **fs_io_s** (or **io_s_**) and dividing by the difference of the sums of the individual **_t_** and **_tu_** values (seconds and microseconds).

   This must be done for a number of samples, with a reasonably small time between samples, in order to get a rate which is reasonably accurate. Since we are sampling the information at a given interval, inaccuracy can exist if the I/O load is not smooth over the sampling time.

   For example, here is a set of samples taken approximately one second apart, when it was known that continuous I/O activity was occurring:

   ```
   _fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862476 _tu_ 634939 _cl_ cluster1.xxx.com
   _fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 3737124864 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 3570 _dir_ 0 _iu_ 5

   _fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862477 _tu_ 645988 _cl_ cluster1.xxx.com
   _fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 3869245440 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 3696 _dir_ 0 _iu_ 5

   _fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862478 _tu_ 647477 _cl_ cluster1.xxx.com
   _fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4120903680 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 3936 _dir_ 0 _iu_ 5

   _fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862479 _tu_ 649363 _cl_ cluster1.xxx.com
   _fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4309647360 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4116 _dir_ 0 _iu_ 5
   ```

```
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862480 _tu_ 650795 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4542431232 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4338 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862481 _tu_ 652515 _cl_ cluster1.ibm.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4743757824 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4530 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862482 _tu_ 654025 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4963958784 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4740 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862483 _tu_ 655782 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 5177868288 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4944 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862484 _tu_ 657523 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 5391777792 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 5148 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862485 _tu_ 665909 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 5599395840 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 5346 _dir_ 0 _iu_ 5
```

This simple **awk** script performs a basic rate calculation:

```
BEGIN {
  count=0;
  prior_t=0;
  prior_tu=0;
  prior_br=0;
  prior_bw=0;
}

{
  count++;

  t = $9;
  tu = $11;
  br = $19;
  bw = $21;

  if(count > 1)
  {
    delta_t = t-prior_t;
    delta_tu = tu-prior_tu;
    delta_br = br-prior_br;
    delta_bw = bw-prior_bw;
    dt = delta_t + (delta_tu / 1000000.0);
    if(dt > 0) {
      rrate = (delta_br / dt) / 1000000.0;
      wrate = (delta_bw / dt) / 1000000.0;

printf("%5.1f MB/sec read %5.1f MB/sec write\n",rrate,wrate);
    }
  }

  prior_t=t;
  prior_tu=tu;
  prior_br=br;
  prior_bw=bw;
}
```

The calculated service rates for each adjacent pair of samples is:

```
  0.0 MB/sec read    130.7 MB/sec write
  0.0 MB/sec read    251.3 MB/sec write
  0.0 MB/sec read    188.4 MB/sec write
  0.0 MB/sec read    232.5 MB/sec write
  0.0 MB/sec read    201.0 MB/sec write
  0.0 MB/sec read    219.9 MB/sec write
  0.0 MB/sec read    213.5 MB/sec write
  0.0 MB/sec read    213.5 MB/sec write
  0.0 MB/sec read    205.9 MB/sec write
```

Since these are discrete samples, there can be variations in the individual results. For example, there may be other activity on the node or interconnection fabric. I/O size, file system block size, and buffering also affect results. There can be many reasons why adjacent values differ. This must be taken into account when building analysis tools that read **mmpmon** output and interpreting results.

For example, suppose a file is read for the first time and gives results like this.

```
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
 92.1 MB/sec read      0.0 MB/sec write
 89.0 MB/sec read      0.0 MB/sec write
 92.1 MB/sec read      0.0 MB/sec write
 90.0 MB/sec read      0.0 MB/sec write
 96.3 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
```

If most or all of the file remains in the GPFS cache, the second read may give quite different rates:

```
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
235.5 MB/sec read      0.0 MB/sec write
287.8 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
```

Considerations such as these need to be taken into account when looking at application I/O service rates calculated from sampling **mmpmon** data.

2. Usage patterns, by sampling at set times of the day (perhaps every half hour) and noticing when the largest changes in I/O volume occur. This does not necessarily give a rate (since there are too few samples) but it can be used to detect peak usage periods.
3. If some nodes service significantly more I/O volume than others over a given time span.
4. When a parallel application is split across several nodes, and is the only significant activity in the nodes, how well the I/O activity of the application is distributed.
5. The total I/O demand that applications are placing on the cluster. This is done by obtaining results from **fs_io_s** and **io_s** in aggregate for all nodes in a cluster.
6. The rate data may appear to be erratic. Consider this example:

```
 0.0 MB/sec read    0.0 MB/sec write
 6.1 MB/sec read    0.0 MB/sec write
92.1 MB/sec read    0.0 MB/sec write
89.0 MB/sec read    0.0 MB/sec write
12.6 MB/sec read    0.0 MB/sec write
 0.0 MB/sec read    0.0 MB/sec write
 0.0 MB/sec read    0.0 MB/sec write
 8.9 MB/sec read    0.0 MB/sec write
92.1 MB/sec read    0.0 MB/sec write
90.0 MB/sec read    0.0 MB/sec write
96.3 MB/sec read    0.0 MB/sec write
 4.8 MB/sec read    0.0 MB/sec write
 0.0 MB/sec read    0.0 MB/sec write
```

The low rates which appear before and after each group of higher rates can be due to the I/O requests occurring late (in the leading sampling period) and ending early (in the trailing sampling period.) This gives an apparently low rate for those sampling periods.

The zero rates in the middle of the example could be caused by reasons such as no I/O requests reaching GPFS during that time period (the application issued none, or requests were satisfied by buffered data at a layer above GPFS), the node becoming busy with other work (causing the application to be undispatched), or other reasons.

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Request histogram (rhist) output - how to aggregate and analyze the results

The **rhist** requests are used to produce histogram data about I/O sizes and latency times for I/O requests.

The output from the **rhist** requests can be used to determine:

1. The number of I/O requests in a given size range. The sizes may vary based on operating system, explicit application buffering, and other considerations. This information can be used to help determine how well an application or set of applications is buffering its I/O. For example, if there are many very small or many very large I/O transactions. A large number of overly small or overly large I/O requests may not perform as well as an equivalent number of requests whose size is tuned to the file system or operating system parameters.
2. The number of I/O requests in a size range that have a given latency time. Many factors can affect the latency time, including but not limited to: system load, interconnection fabric load, file system block size, disk block size, disk hardware characteristics, and the operating system on which the I/O request is issued.

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Using request *source* and prefix directive *once*

The request *source* and prefix directive *once* allow **mmpmon** users to more finely tune their operations.

The **source** request causes **mmpmon** to read requests from a file, and when finished return to reading requests from the input stream.

The prefix directive **once** can be placed in front of any **mmpmon** request. The **once** prefix indicates that the request be run only once, irrespective of the setting of the **-r** flag on the **mmpmon** command. It is useful for requests that do not need to be issued more than once, such as to set up the node list or turn on the request histogram facility.

These rules apply when using the **once** prefix directive and **source** request:

1. **once** with nothing after it is an error that terminates **mmpmon** processing.
2. A file invoked with the **source** request may contain **source** requests, causing file nesting of arbitrary depth. No check is done for loops in this situation.
3. The request **once source** *filename* causes the **once** prefix to be applied to all the **mmpmon** requests in *filename*, including any **source** requests in the file.
4. If a *filename* specified with the **source** request cannot be opened for read, an error is returned and **mmpmon** terminates.
5. If the **-r** flag on the **mmpmon** command has any value other than one, and all requests are prefixed with **once**, **mmpmon** runs all the requests once, issues a message, and then terminates.

An example of *once* and *source* usage

This topic provides and example of the **once** and **source** requests and the output that displays.

This command is issued:

```
mmpmon -p -i command.file -r 0 -d 5000 | tee output.file
```

File **command.file** consists of this:

```
once source mmpmon.header
once rhist nr 512;1024;2048;4096 =
once rhist on
source mmpmon.commands
```

File **mmpmon.header** consists of this:

```
ver
reset
```

File **mmpmon.commands** consists of this:

```
fs_io_s
rhist s
```

The **output.file** is similar to this:

```
_ver_ _n_ 199.18.1.8 _nn_ node1 _v_ 2 _lv_ 4 _vt_ 0
_reset_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770129 _tu_ 511981
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ nr 512;1024;2048;4096 = _rc_ 0 _t_ 1129770131 _tu_ 524674
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ on _rc_ 0 _t_ 1129770131 _tu_ 524921
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770131 _tu_ 525062 _cl_ node1.localdomain
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770131 _tu_ 525062 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770131 _tu_ 525220 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770131 _tu_ 525228 _k_ w
_end_
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770136 _tu_ 526685 _cl_ node1.localdomain
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770136 _tu_ 526685 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 395018 _oc_ 504 _cc_ 252 _rdc_ 0 _wc_ 251 _dir_ 0 _iu_ 147
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770136 _tu_ 526888 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770136 _tu_ 526896 _k_ w
_R_ 0 512 _NR_ 169
_L_ 0.0 1.0 _NL_ 155
_L_ 1.1 10.0 _NL_ 7
_L_ 10.1 30.0 _NL_ 1
_L_ 30.1 100.0 _NL_ 4
_L_ 100.1 200.0 _NL_ 2
_R_ 513 1024 _NR_ 16
_L_ 0.0 1.0 _NL_ 15
_L_ 1.1 10.0 _NL_ 1
_R_ 1025 2048 _NR_ 3
_L_ 0.0 1.0 _NL_ 32
_R_ 2049 4096 _NR_ 18
_L_ 0.0 1.0 _NL_ 18
_R_ 4097 0 _NR_ 16
_L_ 0.0 1.0 _NL_ 16
_end_
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770141 _tu_ 528613 _cl_ node1.localdomain
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770141 _tu_ 528613 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 823282 _oc_ 952 _cc_ 476 _rdc_ 0 _wc_ 474 _dir_ 0 _iu_ 459
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770141 _tu_ 528812 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770141 _tu_ 528820 _k_ w
_R_ 0 512 _NR_ 255
_L_ 0.0 1.0 _NL_ 241
_L_ 1.1 10.0 _NL_ 7
_L_ 10.1 30.0 _NL_ 1
_L_ 30.1 100.0 _NL_ 4
_L_ 100.1 200.0 _NL_ 2
_R_ 513 1024 _NR_ 36
_L_ 0.0 1.0 _NL_ 35
_L_ 1.1 10.0 _NL_ 1
_R_ 1025 2048 _NR_ 90
_L_ 0.0 1.0 _NL_ 90
_R_ 2049 4096 _NR_ 55
_L_ 0.0 1.0 _NL_ 55
_R_ 4097 0 _NR_ 38
_L_ 0.0 1.0 _NL_ 37
_L_ 1.1 10.0 _NL_ 1
_end_
```

```
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770146 _tu_ 530570 _cl_ node1.localdomain
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 1
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770146 _tu_ 530570 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 3069915 _oc_ 1830 _cc_ 914 _rdc_ 0 _wc_ 901 _dir_ 0 _iu_ 1070
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770146 _tu_ 530769 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770146 _tu_ 530778 _k_ w
_R_ 0 512 _NR_ 526
_L_ 0.0 1.0 _NL_ 501
_L_ 1.1 10.0 _NL_ 14
_L_ 10.1 30.0 _NL_ 2
_L_ 30.1 100.0 _NL_ 6
_L_ 100.1 200.0 _NL_ 3
_R_ 513 1024 _NR_ 74
_L_ 0.0 1.0 _NL_ 70
_L_ 1.1 10.0 _NL_ 4
_R_ 1025 2048 _NR_ 123
_L_ 0.0 1.0 _NL_ 117
_L_ 1.1 10.0 _NL_ 6
_R_ 2049 4096 _NR_ 91
_L_ 0.0 1.0 _NL_ 84
_L_ 1.1 10.0 _NL_ 7
_R_ 4097 0 _NR_ 87
_L_ 0.0 1.0 _NL_ 81
_L_ 1.1 10.0 _NL_ 6
_end_
............. and so forth ......................
```

If this command is issued with the same file contents:

```
mmpmon -i command.file -r 0 -d 5000 | tee output.file.english
```

The file **output.file.english** is similar to this:

```
mmpmon node 199.18.1.8 name node1 version 3.1.0
mmpmon node 199.18.1.8 name node1 reset OK
mmpmon node 199.18.1.8 name node1 rhist nr 512;1024;2048;4096 = OK
mmpmon node 199.18.1.8 name node1 rhist on OK
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                   1
timestamp:      1129770175/950895
bytes read:              0
bytes written:           0
opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                   2
timestamp:      1129770175/950895
bytes read:              0
bytes written:

opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770175/951117
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770175/951125
mmpmon node 199.18.1.8 name node1 fs_io_s OK
```

```
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                   1
timestamp:      1129770180/952462
bytes read:              0
bytes written:           0
opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                   2
timestamp:      1129770180/952462
bytes read:              0
bytes written:      491310
opens:                 659
closes:                329
reads:                   0
writes:                327
readdir:                 0
inode updates:          74
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770180/952711
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770180/952720
size range               0 to        512 count        214
  latency range        0.0 to        1.0 count        187
  latency range        1.1 to       10.0 count         15
  latency range       10.1 to       30.0 count          6
  latency range       30.1 to      100.0 count          5
  latency range      100.1 to      200.0 count          1
size range             513 to       1024 count         27
  latency range        0.0 to        1.0 count         26
  latency range      100.1 to      200.0 count          1
size range            1025 to       2048 count         32
  latency range        0.0 to        1.0 count         29
  latency range        1.1 to       10.0 count          1
  latency range       30.1 to      100.0 count          2
size range            2049 to       4096 count         31
  latency range        0.0 to        1.0 count         30
  latency range       30.1 to      100.0 count          1
size range            4097 to          0 count         23
  latency range        0.0 to        1.0 count         23
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                   1
timestamp:      1129770185/954401
bytes read:              0
bytes written:           0
opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                   2
timestamp:      1129770185/954401
bytes read:              0
bytes written:     1641935
```

```
opens:               1062
closes:               531
reads:                  0
writes:               529
readdir:                0
inode updates:        523
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770185/954658
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770185/954667
size range              0 to        512 count       305
  latency range       0.0 to        1.0 count       270
  latency range       1.1 to       10.0 count        21
  latency range      10.1 to       30.0 count         6
  latency range      30.1 to      100.0 count         6
  latency range     100.1 to      200.0 count         2
size range            513 to       1024 count        39
  latency range       0.0 to        1.0 count        36
  latency range       1.1 to       10.0 count         1
  latency range      30.1 to      100.0 count         1
  latency range     100.1 to      200.0 count         1
size range           1025 to       2048 count        89
  latency range       0.0 to        1.0 count        84
  latency range       1.1 to       10.0 count         2
  latency range      30.1 to      100.0 count         3
size range           2049 to       4096 count        56
  latency range       0.0 to        1.0 count        54
  latency range       1.1 to       10.0 count         1
  latency range      30.1 to      100.0 count         1
size range           4097 to          0 count        40
  latency range       0.0 to        1.0 count        39
  latency range       1.1 to       10.0 count         1
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                  1
timestamp:      1129770190/956480
bytes read:             0
bytes written:          0
opens:                  0
closes:                 0
reads:                  0
writes:                 0
readdir:                0
inode updates:          0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                  2
timestamp:      1129770190/956480
bytes read:             0
bytes written:    3357414
opens:               1940
closes:               969
reads:                  0
writes:               952
readdir:                0
inode updates:       1101
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770190/956723
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770190/956732
size range              0 to        512 count       539
  latency range       0.0 to        1.0 count       494
  latency range       1.1 to       10.0 count        29
  latency range      10.1 to       30.0 count         6
  latency range      30.1 to      100.0 count         8
  latency range     100.1 to      200.0 count         2
size range            513 to       1024 count        85
  latency range       0.0 to        1.0 count        81
  latency range        1.1 to       10.0 count          2
  latency range       30.1 to      100.0 count          1
  latency range      100.1 to      200.0 count          1
size range            1025 to       2048 count        133
  latency range        0.0 to        1.0 count        124
  latency range        1.1 to       10.0 count          5
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          3
size range            2049 to       4096 count         99
  latency range        0.0 to        1.0 count         91
  latency range        1.1 to       10.0 count          6
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          1
size range            4097 to          0 count         95
  latency range        0.0 to        1.0 count         90
  latency range        1.1 to       10.0 count          4
  latency range       10.1 to       30.0 count          1
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                  1
timestamp:      1129770195/958310
bytes read:             0
bytes written:          0
opens:                  0
closes:                 0
reads:                  0
writes:                 0
readdir:                0
inode updates:          0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                  2
timestamp:      1129770195/958310
bytes read:             0
bytes written:    3428107
opens:               2046
closes:              1023
reads:                  0
writes:               997
readdir:                0
inode updates:       1321
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770195/958568
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770195/958577
size range               0 to        512 count        555
  latency range        0.0 to        1.0 count        509
  latency range        1.1 to       10.0 count         30
  latency range       10.1 to       30.0 count          6
  latency range       30.1 to      100.0 count          8
  latency range      100.1 to      200.0 count          2
size range             513 to       1024 count         96
  latency range        0.0 to        1.0 count         92
  latency range        1.1 to       10.0 count          2
  latency range       30.1 to      100.0 count          1
  latency range      100.1 to      200.0 count          1
size range            1025 to       2048 count        143
  latency range        0.0 to        1.0 count        134
  latency range        1.1 to       10.0 count          5
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          3
size range            2049 to       4096 count        103
  latency range        0.0 to        1.0 count         95
  latency range        1.1 to       10.0 count          6
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          1
```

```
size range           4097 to          0 count        100
  latency range       0.0 to        1.0 count         95
latency range       1.1 to        10.0 count          4
latency range      10.1 to        30.0 count          1
.............. and so forth ......................
```

For information on interpreting **mmpmon** output results, see the topic *Other information on interpreting mmpmon output*.

Other information about mmpmon output

When interpreting the results from the **mmpmon** output there are several points to consider.

Consider these important points:
- On a node acting as a server of a GPFS file system to NFS clients, NFS I/O is accounted for in the statistics. However, the I/O is that which goes between GPFS and NFS. If NFS caches data, in order to achieve better performance, this activity is not recorded.
- I/O requests made at the application level may not be exactly what is reflected to GPFS. This is dependent on the operating system, and other factors. For example, an application read of 100 bytes may result in obtaining, and caching, a 1 MB block of data at a code level above GPFS (such as the libc I/O layer.) . Subsequent reads within this block result in no additional requests to GPFS.
- The counters kept by **mmpmon** are not atomic and may not be exact in cases of high parallelism or heavy system load. This design minimizes the performance impact associated with gathering statistical data.
- Reads from data cached by GPFS will be reflected in statistics and histogram data. Reads and writes to data cached in software layers above GPFS will be reflected in statistics and histogram data when those layers actually call GPFS for I/O.
- Activity from snapshots affects statistics. I/O activity necessary to maintain a snapshot is counted in the file system statistics.
- Some (generally minor) amount of activity in the root directory of a file system is reflected in the statistics of the file system manager node, and not the node which is running the activity.
- The open count also includes **creat()** call counts.

Counter sizes and counter wrapping

The **mmpmon** command may be run continuously for extended periods of time. The user must be aware that counters may wrap.

This information applies to the counters involved:
- The statistical counters used for the **io_s** and **fs_io_s** requests are maintained by GPFS at all times, even when **mmpmon** has not been invoked. It is suggested that you use the **reset** request prior to starting a sequence of **io_s** or **fs_io_s** requests.
- The bytes read and bytes written counters are unsigned 64-bit integers. They are used in the **fs_io_s** and **io_s** requests, as the **_br_** and **_bw_** fields.
- The counters associated with the **rhist** requests are updated only when the request histogram facility has been enabled.
- The counters used in the **rhist** requests are unsigned 64-bit integers.
- All other counters are unsigned 32-bit integers.

For more information, see the topics *fs_io_s and fs_io output - how to aggregate and analyze the results,* and *Request histogram (rhist) output - how to aggregate and analyze the results.*

Return codes from mmpmon

This topic provides the **mmpmom** return codes and explanations for the codes.

These are the return codes that can appear in the **_rc_** field:

0 Successful completion.

1 One of these has occurred:

1. For the **fs_io_s** request, no file systems are mounted.
2. For an **rhist** request, a request was issued that requires the request histogram facility to be enabled, but it is not. The facility is not enabled if:
   - Since the last **mmstartup** was issued, **rhist on** was never issued.
   - **rhist nr** was issued and **rhist on** was not issued afterwards.

2 For one of the **nlist** requests, the node name is not recognized.

13 For one of the **nlist** requests, the node name is a remote node, which is not allowed.

16 For one of the **rhist** requests, the histogram operations lock is busy. Retry the request.

17 For one of the **nlist** requests, the node name is already in the node list.

22 For one of the **rhist** requests, the size or latency range parameters were not in ascending order or were otherwise incorrect.

233 For one of the **nlist** requests, the specified node is not joined to the cluster.

668 For one of the **nlist** requests, quorum has been lost in the cluster.

Chapter 12. GPFS SNMP support

GPFS supports the use of the SNMP protocol for monitoring the status and configuration of the GPFS cluster. Using an SNMP application, the system administrator can get a detailed view of the system and be instantly notified of important events, such as a node or disk failure.

The Simple Network Management Protocol (SNMP) is an application-layer protocol that facilitates the exchange of management information between network devices. It is part of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth.

SNMP consists of commands to enumerate, read, and write managed variables that are defined for a particular device. It also has a **trap** command, for communicating events asynchronously.

The variables are organized as instances of objects, known as management information bases (MIBs). MIBs are organized in a hierarchical tree by organization (for example, IBM). A GPFS MIB is defined for monitoring many aspects of GPFS.

An SNMP agent software architecture typically consists of a master agent and a set of subagents, which communicate with the master agent through a specific agent/subagent protocol (the AgentX protocol in this case). Each subagent handles a particular system or type of device. A GPFS SNMP subagent is provided, which maps the SNMP objects and their values.

Installing Net-SNMP

The SNMP subagent runs on the collector node of the GPFS cluster. The collector node is designated by the system administrator.

See "Collector node administration" on page 199 for information.

The Net-SNMP master agent (also called the SNMP daemon, or **snmpd**) must be installed on the collector node to communicate with the GPFS subagent and with your SNMP management application. Net-SNMP is included in most Linux distributions and should be supported by your Linux vendor. Source and binaries for several platforms are available from the download section of the Net-SNMP website (www.net-snmp.org/download.html).

Note: Currently, the collector node must run on the Linux operating system. For an up-to-date list of supported operating systems, specific distributions, and other dependencies, refer to the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

The GPFS subagent expects to find the following shared object libraries:

```
libnetsnmpagent.so       -- from Net-SNMP
libnetsnmphelpers.so     -- from Net-SNMP
libnetsnmpmibs.so        -- from Net-SNMP
libnetsnmp.so            -- from Net-SNMP
libwrap.so               -- from TCP Wrappers
libcrypto.so             -- from OpenSSL
```

Note: TCP Wrappers and OpenSSL are prerequisites and should have been installed when you installed Net-SNMP.

The installed libraries will be found in **/lib64** or **/usr/lib64** or **/usr/local/lib64**. They may be installed under names like **libnetsnmp.so.5.1.2**. The GPFS subagent expects to find them without the appended version information in the name. Library installation should create these symbolic links for you, so you will rarely need to create them yourself. You can ensure that symbolic links exist to the versioned name from the plain name. For example,

```
# cd /usr/lib64
# ln -s libnetsnmpmibs.so.5.1.2 libnetsnmpmibs.so
```

Repeat this process for all the libraries listed in this topic.

Note: For possible Linux platform and Net-SNMP version compatibility restrictions, see the GPFS README and the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Configuring Net-SNMP

The GPFS subagent process connects to the Net-SNMP master agent, **snmpd**.

The following entries are required in the **snmpd** configuration file on the collector node (usually, **/etc/snmp/snmpd.conf**):

```
master agentx
AgentXSocket tcp:localhost:705
trap2sink managementhost
```

where:

managementhost
: Is the host name or IP address of the host to which you want SNMP traps sent.

If your GPFS cluster has a large number of nodes or a large number of file systems for which information must be collected, you must increase the timeout and retry parameters for communication between the SNMP master agent and the GPFS subagent to allow time for the volume of information to be transmitted. The **snmpd** configuration file entries for this are:

```
agentXTimeout 60
agentXRetries 10
```

where:

agentXTimeout
: Is set to 60 seconds for subagent to master agent communication.

agentXRetries
: Is set to 10 for the number of communication retries.

Note: Other values may be appropriate depending on the number of nodes and file systems in your GPFS cluster.

After modifying the configuration file, restart the SNMP daemon.

Configuring management applications

To configure any SNMP-based management applications you might be using (such as Tivoli NetView® or Tivoli Netcool®, or others), you must make the GPFS MIB file available on the processor on which the management application runs.

You must also supply the management application with the host name or IP address of the collector node to be able to extract GPFS monitoring information through SNMP. To do this, you must be familiar with your SNMP-based management applications.

For more information about Tivoli NetView or Tivoli Netcool, see IBM Knowledge Center (www.ibm.com/support/knowledgecenter).

Installing MIB files on the collector node and management node

The GPFS management information base (MIB) file is found on the collector node in the **/usr/lpp/mmfs/data** directory with the name **GPFS-MIB.txt**.

To install this file on the collector node, do the following:

1. Copy or link the **/usr/lpp/mmfs/data/GPFS-MIB.txt** MIB file into the **SNMP MIB** directory (usually, **/usr/share/snmp/mibs**).

   Alternatively, you could add the following line to the **snmp.conf** file (usually found in the directory **/etc/snmp**):
   ```
   mibdirs +/usr/lpp/mmfs/data
   ```
2. Add the following entry to the **snmp.conf** file (usually found in the directory **/etc/snmp**):
   ```
   mibs +GPFS-MIB
   ```
3. Restart the SNMP daemon.

Different management applications have different locations and ways for installing and loading a new MIB file. The following steps for installing the GPFS MIB file apply only to Net-SNMP. If you are using other management applications, such as NetView and NetCool, refer to corresponding product manuals (listed in "Configuring management applications" on page 198) for the procedure of MIB file installation and loading.

1. Remotely copy the **/usr/lpp/mmfs/data/GPFS-MIB.txt** MIB file from the collector node into the **SNMP MIB** directory (usually, **/usr/share/snmp/mibs**).
2. Add the following entry to the **snmp.conf** file (usually found in the directory **/etc/snmp**):
   ```
    mibs +GPFS-MIB
   ```
3. You might need to restart the SNMP management application. Other steps might be necessary to make the GPFS MIB available to your management application.

Collector node administration

Collector node administration includes: assigning, unassigning, and changing collector nodes. You can also see if a collector node is defined.

To assign a collector node and start the SNMP agent, enter:

```
mmchnode --snmp-agent -N NodeName
```

To unassign a collector node and stop the SNMP agent, enter:

```
mmchnode --nosnmp-agent -N NodeName
```

To see if there is a GPFS SNMP subagent collector node defined, enter:

```
mmlscluster | grep snmp
```

To change the collector node, issue the following two commands:

```
mmchnode --nosnmp-agent -N OldNodeName

mmchnode --snmp-agent -N NewNodeName
```

Starting and stopping the SNMP subagent

The SNMP subagent is started and stopped automatically.

The SNMP subagent is started automatically when GPFS is started on the collector node. If GPFS is already running when the collector node is assigned, the **mmchnode** command will automatically start the SNMP subagent.

The SNMP subagent is stopped automatically when GPFS is stopped on the node (**mmshutdown**) or when the SNMP collector node is unassigned (**mmchnode**).

The management and monitoring subagent

The GPFS SNMP management and monitoring subagent runs under an SNMP master agent such as Net-SNMP. It handles a portion of the SNMP OID space.

The management and monitoring subagent connects to the GPFS daemon on the collector node to retrieve updated information about the status of the GPFS cluster.

SNMP data can be retrieved using an SNMP application such as Tivoli NetView. NetView provides a MIB browser for retrieving user-requested data, as well as an event viewer for displaying asynchronous events.

Information that is collected includes status, configuration, and performance data about GPFS clusters, nodes, disks, file systems, storage pools, and asynchronous events. The following is a sample of the data that is collected for each of the following categories:

- Cluster status and configuration (see "Cluster status information" on page 201 and "Cluster configuration information" on page 201)
  - Name
  - Number of nodes
  - Primary and secondary servers
- Node status and configuration (see "Node status information" on page 202 and "Node configuration information" on page 202)
  - Name
  - Current status
  - Type
  - Platform
- File system status and performance (see "File system status information" on page 203 and "File system performance information" on page 204)
  - Name
  - Status
  - Total space
  - Free space
  - Accumulated statistics
- Storage pools (see "Storage pool information" on page 204)
  - Name
  - File system to which the storage pool belongs
  - Total storage pool space
  - Free storage pool space
  - Number of disks in the storage pool
- Disk status, configuration, and performance (see "Disk status information" on page 205, "Disk configuration information" on page 205, and "Disk performance information" on page 206)
  - Name
  - Status
  - Total space

  - Free space
  - Usage (metadata/data)
  - Availability
  - Statistics
- Asynchronous events (traps) (see "Net-SNMP traps" on page 207)
  - File system mounted or unmounted
  - Disks added, deleted, or changed
  - Node failure or recovery
  - File system creation, deletion, or state change
  - Storage pool is full or nearly full

Note: If file systems are not mounted on the collector node at the time that an SNMP request is received, the subagent can still obtain a list of file systems, storage pools, and disks, but some information, such as performance statistics, will be missing.

SNMP object IDs

This topic defines the SNMP object IDs.

The management and monitoring SNMP subagent serves the OID space defined as **ibm.ibmProd.ibmGPFS**, which is the numerical **enterprises.2.6.212** OID space.

Underneath this top-level space are the following:

- **gpfsTraps** at **ibmGPFS.0**
- **gpfsMIBObjects** at **ibmGPFS.1**

MIB objects

gpfsMIBObjects provides a space of objects that can be retrieved using a MIB browser application. Net-SNMP provides the **snmpget**, **snmpgetnext**, **snmptable**, and **snmpwalk** commands, which can be used to retrieve the contents of these fields.

Cluster status information

This topic provides the values and descriptions for the GPFS cluster.

Table 31 shows the current status information for the GPFS cluster:

Table 31. gpfsClusterStatusTable: Cluster status information

Value	Description
gpfsClusterName	The cluster name.
gpfsClusterId	The cluster ID.
gpfsClusterMinReleaseLevel	The currently enabled cluster functionality level.
gpfsClusterNumNodes	The number of nodes that belong to the cluster.
gpfsClusterNumFileSystems	The number of file systems that belong to the cluster.

Cluster configuration information

This topic shows the values and descriptions for the GPFS cluster configuration.

Table 32 on page 202 shows the GPFS cluster configuration information:

Table 32. gpfsClusterConfigTable: Cluster configuration information

Value	Description
gpfsClusterConfigName	The cluster name.
gpfsClusterUidDomain	The UID domain name for the cluster.
gpfsClusterRemoteShellCommand	The remote shell command being used.
gpfsClusterRemoteFileCopyCommand	The remote file copy command being used.
gpfsClusterPrimaryServer	The primary GPFS cluster configuration server.
gpfsClusterSecondaryServer	The secondary GPFS cluster configuration server.
gpfsClusterMaxBlockSize	The maximum file system block size.
gpfsClusterDistributedTokenServer	Indicates whether the distributed token server is enabled.
gpfsClusterFailureDetectionTime	The desired time for GPFS to react to a node failure.
gpfsClusterTCPPort	The TCP port number.
gpfsClusterMinMissedPingTimeout	The lower bound on a missed **ping** timeout (seconds).
gpfsClusterMaxMissedPingTimeout	The upper bound on missed **ping** timeout (seconds).

Node status information

This topic provides a description for each GPFS node.

Table 33 shows the collected status data for each node:

Table 33. gpfsNodeStatusTable: Node status information

Node	Description
gpfsNodeName	The node name used by the GPFS daemon.
gpfsNodeIp	The node IP address.
gpfsNodePlatform	The operating system being used.
gpfsNodeStatus	The node status (for example, up or down).
gpfsNodeFailureCount	The number of node failures.
gpfsNodeThreadWait	The longest hung thread's wait time (milliseconds).
gpfsNodeHealthy	Indicates whether the node is healthy in terms of hung threads. If there are hung threads, the value is no.
gpfsNodeDiagnosis	Shows the number of hung threads and detail on the longest hung thread.
gpfsNodeVersion	The GPFS product version of the currently running daemon.

Node configuration information

This topic shows the collected configuration data for each GPFS node.

Table 34 shows the collected configuration data for each node:

Table 34. gpfsNodeConfigTable: Node configuration information

Node	Description
gpfsNodeConfigName	The node name used by the GPFS daemon.
gpfsNodeType	The node type (for example, manager/client or quorum/nonquorum).

Table 34. gpfsNodeConfigTable: Node configuration information (continued)

Node	Description
gpfsNodeAdmin	Indicates whether the node is one of the preferred admin nodes.
gpfsNodePagePoolL	The size of the cache (low 32 bits).
gpfsNodePagePoolH	The size of the cache (high 32 bits).
gpfsNodePrefetchThreads	The number of prefetch threads.
gpfsNodeMaxMbps	An estimate of how many megabytes of data can be transferred per second.
gpfsNodeMaxFilesToCache	The number of inodes to cache for recently-used files that have been closed.
gpfsNodeMaxStatCache	The number of inodes to keep in the stat cache.
gpfsNodeWorker1Threads	The maximum number of worker threads that can be started.
gpfsNodeDmapiEventTimeout	The maximum time the file operation threads will block while waiting for a DMAPI synchronous event (milliseconds).
gpfsNodeDmapiMountTimeout	The maximum time that the mount operation will wait for a disposition for the mount event to be set (seconds).
gpfsNodeDmapiSessFailureTimeout	The maximum time the file operation threads will wait for the recovery of the failed DMAPI session (seconds).
gpfsNodeNsdServerWaitTimeWindowOnMount	Specifies a window of time during which a mount can wait for NSD servers to come up (seconds).
gpfsNodeNsdServerWaitTimeForMount	The maximum time that the mount operation will wait for NSD servers to come up (seconds).
gpfsNodeUnmountOnDiskFail	Indicates how the GPFS daemon will respond when a disk failure is detected. If it is "true", any disk failure will cause only the local node to forcibly unmount the file system that contains the failed disk.

File system status information

This topic shows the collected status information for each GPFS file system.

Table 35 shows the collected status information for each file system:

Table 35. gpfsFileSystemStatusTable: File system status information

Value	Description
gpfsFileSystemName	The file system name.
gpfsFileSystemStatus	The status of the file system.
gpfsFileSystemXstatus	The executable status of the file system.
gpfsFileSystemTotalSpaceL	The total disk space of the file system in kilobytes (low 32 bits).
gpfsFileSystemTotalSpaceH	The total disk space of the file system in kilobytes (high 32 bits).
gpfsFileSystemNumTotalInodesL	The total number of file system inodes (low 32 bits).
gpfsFileSystemNumTotalInodesH	The total number of file system inodes (high 32 bits).
gpfsFileSystemFreeSpaceL	The free disk space of the file system in kilobytes (low 32 bits).

Table 35. gpfsFileSystemStatusTable: File system status information (continued)

Value	Description
gpfsFileSystemFreeSpaceH	The free disk space of the file system in kilobytes (high 32 bits).
gpfsFileSystemNumFreeInodesL	The number of free file system inodes (low 32 bits).
gpfsFileSystemNumFreeInodesH	The number of free file system inodes (high 32 bits).

File system performance information

This topic provides the GPFS file system performance information.

Table 36 shows the file system performance information:

Table 36. gpfsFileSystemPerfTable: File system performance information

Value	Description
gpfsFileSystemPerfName	The file system name.
gpfsFileSystemBytesReadL	The number of bytes read from disk, not counting those read from cache (low 32 bits).
gpfsFileSystemBytesReadH	The number of bytes read from disk, not counting those read from cache (high 32 bits).
gpfsFileSystemBytesCacheL	The number of bytes read from the cache (low 32 bits).
gpfsFileSystemBytesCacheH	The number of bytes read from the cache (high 32 bits).
gpfsFileSystemBytesWrittenL	The number of bytes written, to both disk and cache (low 32 bits).
gpfsFileSystemBytesWrittenH	The number of bytes written, to both disk and cache (high 32 bits).
gpfsFileSystemReads	The number of read operations supplied from disk.
gpfsFileSystemCaches	The number of read operations supplied from cache.
gpfsFileSystemWrites	The number of write operations to both disk and cache.
gpfsFileSystemOpenCalls	The number of file system open calls.
gpfsFileSystemCloseCalls	The number of file system close calls.
gpfsFileSystemReadCalls	The number of file system read calls.
gpfsFileSystemWriteCalls	The number of file system write calls.
gpfsFileSystemReaddirCalls	The number of file system readdir calls.
gpfsFileSystemInodesWritten	The number of inode updates to disk.
gpfsFileSystemInodesRead	The number of inode reads.
gpfsFileSystemInodesDeleted	The number of inode deletions.
gpfsFileSystemInodesCreated	The number of inode creations.
gpfsFileSystemStatCacheHit	The number of stat cache hits.
gpfsFileSystemStatCacheMiss	The number of stat cache misses.

Storage pool information

This topic provides the collected information for each GPFS storage pool.

Table 37 on page 205 shows the collected information for each storage pool:

Table 37. gpfsStgPoolTable: Storage pool information

Value	Description
gpfsStgPoolName	The name of the storage pool.
gpfsStgPoolFSName	The name of the file system to which the storage pool belongs.
gpfsStgPoolTotalSpaceL	The total disk space in the storage pool in kilobytes (low 32 bits).
gpfsStgPoolTotalSpaceH	The total disk space in the storage pool in kilobytes (high 32 bits).
gpfsStgPoolFreeSpaceL	The free disk space in the storage pool in kilobytes (low 32 bits).
gpfsStgPoolFreeSpaceH	The free disk space in the storage pool in kilobytes (high 32 bits).
gpfsStgPoolNumDisks	The number of disks in the storage pool.

Disk status information

This topic provides the collected status information for each GPFS disk.

Table 38 shows the collected status information for each disk:

Table 38. gpfsDiskStatusTable: Disk status information

Value	Description
gpfsDiskName	The disk name.
gpfsDiskFSName	The name of the file system to which the disk belongs.
gpfsDiskStgPoolName	The name of the storage pool to which the disk belongs.
gpfsDiskStatus	The status of a disk (values: NotInUse, InUse, Suspended, BeingFormatted, BeingAdded, To Be Emptied, Being Emptied, Emptied, BeingDeleted, BeingDeleted-p, ReferencesBeingRemoved, BeingReplaced or Replacement).
gpfsDiskAvailability	The availability of the disk (Unchanged, OK, Unavailable, Recovering).
gpfsDiskTotalSpaceL	The total disk space in kilobytes (low 32 bits).
gpfsDiskTotalSpaceH	The total disk space in kilobytes (high 32 bits).
gpfsDiskFullBlockFreeSpaceL	The full block (unfragmented) free space in kilobytes (low 32 bits).
gpfsDiskFullBlockFreeSpaceH	The full block (unfragmented) free space in kilobytes (high 32 bits).
gpfsDiskSubBlockFreeSpaceL	The sub-block (fragmented) free space in kilobytes (low 32 bits).
gpfsDiskSubBlockFreeSpaceH	The sub-block (fragmented) free space in kilobytes (high 32 bits).

Disk configuration information

This topic provides the collected configuration information for each GPFS disk.

Table 39 on page 206 shows the collected disk configuration information for each disk:

Table 39. gpfsDiskConfigTable: Disk configuration information

Value	Description
gpfsDiskConfigName	The disk name.
gpfsDiskConfigFSName	The name of the file system to which the disk belongs.
gpfsDiskConfigStgPoolName	The name of the storage pool to which the disk belongs.
gpfsDiskMetadata	Indicates whether the disk holds metadata.
gpfsDiskData	Indicates whether the disk holds data.

Disk performance information

This topic provides the collected configuration information for each disk.

Table 40 shows the collected disk performance information for each disk:

Table 40. gpfsDiskPerfTable: Disk performance information

Value	Description
gpfsDiskPerfName	The disk name.
gpfsDiskPerfFSName	The name of the file system to which the disk belongs.
gpfsDiskPerfStgPoolName	The name of the storage pool to which the disk belongs.
gpfsDiskReadTimeL	The total time spent waiting for disk read operations (low 32 bits).
gpfsDiskReadTimeH	The total time spent waiting for disk read operations (high 32 bits).
gpfsDiskWriteTimeL	The total time spent waiting for disk write operations in microseconds (low 32 bits).
gpfsDiskWriteTimeH	The total time spent waiting for disk write operations in microseconds (high 32 bits).
gpfsDiskLongestReadTimeL	The longest disk read time in microseconds (low 32 bits).
gpfsDiskLongestReadTimeH	The longest disk read time in microseconds (high 32 bits).
gpfsDiskLongestWriteTimeL	The longest disk write time in microseconds (low 32 bits).
gpfsDiskLongestWriteTimeH	The longest disk write time in microseconds (high 32 bits).
gpfsDiskShortestReadTimeL	The shortest disk read time in microseconds (low 32 bits).
gpfsDiskShortestReadTimeH	The shortest disk read time in microseconds (high 32 bits).
gpfsDiskShortestWriteTimeL	The shortest disk write time in microseconds (low 32 bits).
gpfsDiskShortestWriteTimeH	The shortest disk write time in microseconds (high 32 bits).
gpfsDiskReadBytesL	The number of bytes read from the disk (low 32 bits).
gpfsDiskReadBytesH	The number of bytes read from the disk (high 32 bits).
gpfsDiskWriteBytesL	The number of bytes written to the disk (low 32 bits).
gpfsDiskWriteBytesH	The number of bytes written to the disk (high 32 bits).
gpfsDiskReadOps	The number of disk read operations.

Table 40. gpfsDiskPerfTable: Disk performance information (continued)

Value	Description
gpfsDiskWriteOps	The number of disk write operations.

Net-SNMP traps

Traps provide asynchronous notification to the SNMP application when a particular event has been triggered in GPFS.

Table 41 shows the trap types that are defined:

Table 41. Net-SNMP traps

Net-SNMP trap type	This event is triggered by:
Mount	By the mounting node when the file system is mounted on a node.
Unmount	By the unmounting node when the file system is unmounted on a node.
Add Disk	By the file system manager when a disk is added to a file system on a node.
Delete Disk	By the file system manager when a disk is deleted from a file system.
Change Disk	By the file system manager when the status of a disk or the availability of a disk is changed within the file system.
SGMGR Takeover	By the cluster manager when a file system manager takeover is successfully completed for the file system.
Node Failure	By the cluster manager when a node fails.
Node Recovery	By the cluster manager when a node recovers normally.
File System Creation	By the file system manager when a file system is successfully created.
File System Deletion	By the file system manager when a file system is deleted.
File System State Change	By the file system manager when the state of a file system changes.
New Connection	When a new connection thread is established between the events exporter and the management application.
Event Collection Buffer Overflow	By the collector node when the internal event collection buffer in the GPFS daemon overflows.
Hung Thread	By the affected node when a hung thread is detected. The GPFS Events Exporter Watchdog thread periodically checks for threads that have been waiting for longer than a threshold amount of time.
Storage Pool Utilization	By the file system manager when the utilization of a storage pool becomes full or almost full.

Chapter 13. Identity management on Windows

GPFS allows file sharing among AIX, Linux, and Windows nodes. AIX and Linux rely on 32-bit user and group IDs for file ownership and access control purposes, while Windows uses variable-length security identifiers (SIDs). The difference in the user identity description models presents a challenge to any subsystem that allows for heterogeneous file sharing.

GPFS uses 32-bit ID namespace as the canonical namespace, and Windows SIDs are mapped into this namespace as needed. Two different mapping algorithms are used (depending on system configuration):

- GPFS built-in auto-generated mapping
- User-defined mappings stored in the Microsoft Windows Active Directory using the Microsoft Identity Management for UNIX (IMU) component

Auto-generated ID mappings

Auto-generated ID mappings are the default. If no explicit mappings are created by the system administrator in the Active Directory using Microsoft Identity Management for UNIX (IMU), all mappings between security identifiers (SIDs) and UNIX IDs will be created automatically using a reserved range in UNIX ID space.

Note: If you have a mix of GPFS running on Windows and other Windows clients accessing the integrated SMB server function, the ability to share data between these clients has not been tested or validated. With protocol support, the SMB server may also be configured to automatically generate ID mapping. If you want to ensure that SMB users do not access data (share ID mapping) with Windows users, ensure that the automatic range for SMB server is different from this range. The range of IDs automatically generated for the SMB server can be controlled by **mmuserauth**.

Unless the default reserved ID range overlaps with an ID already in use, no further configuration is needed to use the auto-generated mapping function. If you have a specific file system or subtree that are only accessed by user applications from Windows nodes (even if AIX or Linux nodes are used as NSD servers), auto-generated mappings will be sufficient for all application needs.

The default reserved ID range used by GPFS starts with ID 15,000,000 and covers 15,000,000 IDs. The reserved range should not overlap with any user or group ID in use on any AIX or Linux nodes. To change the starting location or the size of the reserved ID range, use the following GPFS configuration parameters:

`sidAutoMapRangeLength`
Controls the length of the reserved range for Windows SID to UNIX ID mapping.

`sidAutoMapRangeStart`
Specifies the start of the reserved range for Windows SID to UNIX ID mapping.

Note: For planning purposes, remember that auto-generated ID mappings are stored permanently with file system metadata. A change in the **sidAutoMapRangeStart** value is only effective for file systems created after the configuration change.

Installing Windows IMU

The Identity Management for UNIX (IMU) feature is included in Windows Server. This feature needs to be installed on the primary domain controller, as well as on any backup domain controllers. It is not installed by default. There are two components that need to be installed in order for IMU to function correctly.

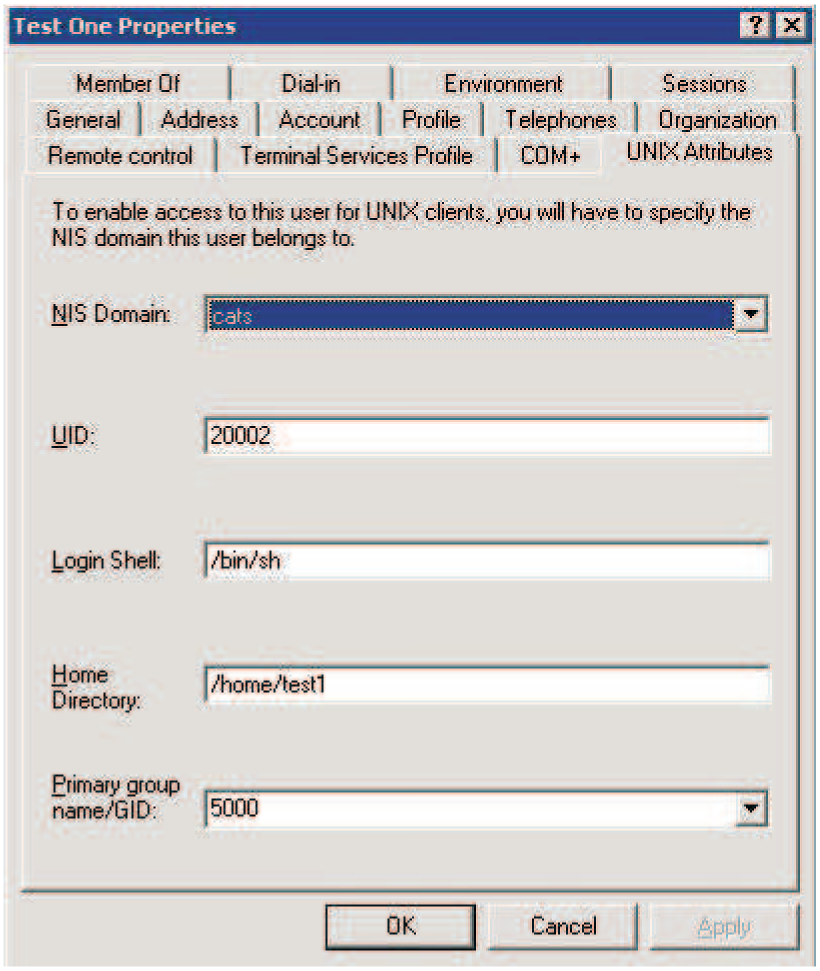
When Active Directory is running on Windows Server 2008, follow these steps to add the IMU service:

1. Open Server Manager.
2. Under Roles, select **Active Directory Domain Services**.
3. Under Role Services, select **Add Role Services**.
4. Under the Identity Management for UNIX role service, check **Server for Network Information Services**.
5. Click **Next**, then **Install**.
6. Restart the system when the installation completes.

Configuring ID mappings in IMU

To configure ID mappings in Microsoft Identity Management for UNIX (IMU), follow the steps in this procedure.

1. Open **Active Directory Users and Computers** (accessible under Administrative Tools).
2. Select the **Users** branch in the tree on the left under the branch for your domain to see the list of users and groups in this domain.
3. Double-click on any user or group line to bring up the Properties window. If IMU is set up correctly, there will be a **UNIX Attributes** tab as shown in Figure 11:



To enable access to this user for UNIX clients, you will have to specify the NIS domain this user belongs to.

NIS Domain: cats

UID: 20002

Login Shell: /bin/sh

Home Directory: /home/test1

Primary group name/GID: 5000

Figure 11. Properties window

Note: Because the IMU subsystem was originally designed to support integration with the UNIX Network Information Service (NIS), there is an **NIS Domain** field in the Properties window. You do not need to have NIS set up on the UNIX side. For GPFS, the NIS language does not matter.

Update information on the **UNIX Attributes** panel as follows:
1. Under the **NIS Domain** drop-down list, select the name of your Active Directory domain. Selecting **<none>** will remove an existing mapping.
2. Specify a UID in the **UID** field, and for Group objects, specify a GID. This will create a bidirectional mapping between the corresponding SID and a UNIX ID. IMU will disallow the use of the same UID or GID for more than one user or group to ensure that all mappings are unique. In addition to creating mappings for domain users and groups, you can create mappings for certain built-in accounts by going to the **Builtin branch** in the Active Directory Users and Computers panel.
3. Disregard the **Primary group name/GID** field because GPFS does not use it.

It is generally better to configure all ID mappings before mounting a GPFS file system for the first time. Doing that ensures that GPFS only stores properly remapped IDs on disk. However, it is possible to add or delete mappings at any time while GPFS file systems are mounted. GPFS picks up mapping changes dynamically (the code currently checks for mapping changes every 60 seconds), and will start using them at that time.

If an IMU mapping is configured for an ID that is already recorded in some file metadata, you must proceed with caution to avoid user confusion and access disruption. Auto-generated mappings already stored in access control lists (ACLs) on disk will continue to map correctly to Windows SIDs. However, because the SID is now mapped to a different UNIX ID, when you access a file with an ACL containing its auto-generated ID, this access will effectively appear to GPFS as an access by a different user. Depending on the file access permissions, you might not be able to access files that were previously accessible. Rewriting affected ACLs after setting up a new mapping will help replace auto-generated IDs with IMU-mapped IDs, and will restore proper file access for the affected ID (this operation might need to be performed by the system administrator). Examining file ownership and permission information from a UNIX node (for example, using the **mmgetacl** command) is the easiest way to determine whether the ACL for a specified file contains auto-generated or IMU-mapped IDs.

Chapter 14. Active file management

Active file management (AFM) is a scalable, high-performance, file system caching layer integrated with the GPFS cluster file system. AFM allows you to create associations from a local GPFS cluster to a remote cluster or storage, and to define the location and flow of file data to automate the management of the data. This allows you to implement a single namespace view across sites around the world.

Note: This feature is available with IBM Spectrum Scale Standard Edition or higher.

AFM masks wide-area network latencies and outages by using GPFS to cache massive data sets, allowing data access and modifications even when remote storage cluster is unavailable. In addition, AFM performs updates to the remote cluster asynchronously, which allows applications to continue operating while not being constrained by limited outgoing network bandwidth.

The AFM implementation leverages the inherent scalability of GPFS to provide a multinode, consistent cache of data located at a home cluster. By integrating with the file system, AFM provides a POSIX-compliant interface, making the cache completely transparent to applications. AFM is easy to deploy, as it relies on open standards for high-performance file serving and does not require any proprietary hardware or software to be installed at the home cluster.

By using NFSv3 or GPFS protocol to cache data, AFM can improve network performance to any home cluster. The performance of AFM is limited by NFS whenever the NFS protocol is used.

This topic provides an overview of technical details, while the restrictions are listed in the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Active file management architecture

Active file management (AFM) uses a home-and-cache model in which a single home provides the primary storage of data, and exported data is cached in a local GPFS file system.

Home Home can be an NFS export from a remote cluster. The export point can be a local file system in the remote cluster, a GPFS file system or a GPFS fileset in the remote cluster. AFM uses a proprietary protocol over NFS.

In addition, AFM is supported when a remote file system is mounted on the cache cluster using GPFS protocol. Native GPFS protocol utilizes remote file system mount based over a multicluster configuration to function as the AFM target. This requires that a multicluster setup exist between the home and cache before AFM can use the home cluster's file system mount on the remote cluster for AFM operations.

Architecturally, AFM works with any file system at the home cluster; however, ACLs, extended attributes, and sparse files are only supported when the home file system is GPFS, irrespective of whether NFS or GPFS target is used. The **mmafmconfig** command should be run on the home cluster to enable this support.

Cache The container used to cache home data is a GPFS fileset. Each AFM-enabled fileset has a single home cluster associated with it (represented by the hostname of the home server).

Each cache fileset in a cluster is served by one of the nodes designated as gateway in the cluster. The gateway node mapped to serve a fileset is called the metadata server (MDS) of the fileset. The MDS acts as the owner for the fileset. All other nodes in the cluster, including other gateways, become application nodes for the fileset. A fileset can have multiple application nodes (which service application data requests). All other gateway nodes can be configured to help the MDS in fetching data to and from the home cluster. See “Parallel I/O” on page 227.

The split between application and gateway nodes is conceptual, and any node in the cache cluster can function as both a gateway node and an application node based on its configuration. The gateway nodes can be viewed as the edge of the cache cluster that can communicate with the home cluster, while the application nodes interface with the application. Gateway and application nodes communicate with each other via internal RPC requests.

Any cluster can be a home cluster, a cache cluster, or both. In typical usage, a home would be in one GPFS cluster and a cache would be defined in another, but this is not required. In fact, a cluster can be a home for one fileset and can be a cache for another fileset; and multiple AFM-enabled filesets may be defined in one cache, each caching from a different home cluster. This provides great flexibility in how you can leverage the caching behavior.

A cache can request data by reading a file or by pre-fetching the data. Any time a file is read, if the file data is not yet in the cache or is not up to date, the data is copied from the home into the cache.

Multiple cache filesets can read data from a single home. In single-writer mode, only one cache can write data to a single home. In independent-writer (IW) mode, multiple caches can write to a single home, as long as each cache writes to different files. In case multiple caches write to same file, the sequence of updates is nondeterministic.

AFM filesets can cache extended attributes and ACLs. To enable this functionality, the home needs to issue the **mmafmconfig** command.

Notes:
1. AFM uses certain internal directories, such as the following, which must not be altered or removed:
   - `.afm` at home
   - `.ptrash` (see “Failover of cache filesets” on page 225
   - `.pconflicts` in cache (see “Failover of cache filesets” on page 225)
2. User IDs and group IDs must be managed the same way across cache and home.
3. For AFM relationships, which are using native GPFS protocol, where user ids are different on the home and the cache, the ids may be remapped using GPFS UID remapping.

Caching modes

Active file management (AFM) caching modes control the flow of data.

The following table shows the AFM caching modes that are available.

Table 42. AFM modes

Mode	Description
Local-update (LU)	Cached data (data from home) is read-only. The difference from read-only mode is that you can create and modify files in the cache fileset. Files that are created or modified in the cache are considered local updates. Local updates are never pushed back to home. Once a file is modified in the cache, the file is no longer compared to the version at home to verify that it is up to date. Appending to, truncating, or writing to an uncached file fetches the entire file to the cache before making the change locally. **Note:** Any small change done in the LU fileset directory could cause the fileset to be marked as local and lose context with the home. For example, running **chmod** or **chown** of the LU fileset root directory causes the entire fileset to be out of sync with the home.

Table 42. AFM modes (continued)

Mode	Description
Read-only (RO)	Data in the cache is read-only. Creating or modifying files in the cache fileset is not allowed.
Single-writer (SW)	One cache fileset does all the writing. The assumption is that all of the other cache filesets associated with the same home are configured as read-only or local-update. This mode avoids possible write conflicts. There is no explicit enforcement of a single-writer mode in GPFS. Rather, the administrator needs to guarantee that there are no other writeable caches associated with this home and that no writes are done at home. On single writer filesets, a write at home is not allowed. However, if there is a write/corruption at home, inadvertently or due to some error situation, it results in a conflicted scenario for the single-writer fileset. The administrator-monitored command **resync** can be used to recover home. When running **resync**, the administrator must ensure that the fileset is in **Active** or **NeedsResync** state. Cache can sometimes detect inconsistencies and put the fileset in **NeedsResync** state. This implies that **resync** gets run automatically during the next request to the gateway, and the inconsistencies are resolved. If they are not resolved automatically, they must be manually resolved by running the **resync** command. Appending to or truncating a file in SW mode does not fetch the file into cache, but queues it to home.
Independent-writer (IW)	Allows multiple caches to write to different files. There is no locking between clusters. Each cache makes its updates to home independently. Independent writer mode supports transparent synchronization with home. That is, changes in IW cache are synchronized with home and vice versa. In case data is updated at home, all connected IW caches fetch those changes on demand based on the revalidation intervals set. Multiple active cache filesets point to one home file system or fileset and can modify the data at the same time. This mode must only be used by applications running in caches that do not require synchronization to properly update the data on the home cluster. The multiple cache sites do revalidation periodically and pull the new data from home. This mode is typically used to access different files from each IW cache site (for example, unique users at each site updating files in their home directory). While this mode allows multiple cache clusters to modify the same set of files, this must be only done by advanced users. This is because there is no locking or ordering between updates; the updates are propagated to the home in an asynchronous manner and might be delayed due to network disconnections. Therefore, conflicting updates from multiple cache sites can cause the data at the home site to be undetermined. Appending to or truncating a file in IW mode does not fetch the file into cache, but queues it to home.

Notes:

- SW and IW cache filesets must not access the same data; if they do, serious problems can occur. Only RO, LU, and IW cache filesets should access the same home dataset.
- Resync cannot be used with IW filesets, as multiple cache filesets can update the same set of data.
- Before converting from IW to SW mode, ensure that all the remaining IWs are converted to either RO or LU to avoid the risk of home conflicts caused by writes from other IWs. When a large number of IW caches point to the same home, the number of NFS threads at home might be tuned for better efficiency. Also, if the data updates are unique, increasing the revalidation timers for the IW caches might reduce the frequency of periodic refreshes from home and improve cache performance.

- Revalidations are per node, not per fileset. If a fileset is revalidated from one node on the cache cluster, the same fileset goes through another revalidation when accessed from another node on the cache cluster.

File system caching and synchronization

AFM treats all file system operations as either asynchronous or synchronous.

Asynchronous operations

When an asynchronous operation is used, an application can proceed as soon as the request is queued on the gateway node. An MDS of a fileset can delay asynchronous operations for the time defined by the synchronization lag (this can be configured using the **mmchfileset** command).

Files and directories are refreshed from the home cluster based on a validity lag, and file system updates are synchronized to the home cluster based on the synchronization lag.

Within a single cache cluster, application nodes will experience POSIX semantics.

Data and metadata updates

All operations that modify the file system are synchronized to the home server at an interval determined by the synchronization lag. It is important to note that updates might be sent to the home cluster much sooner than the lag if proceeding operations depend on their synchronization for correctness.

AFM sends data from cache to home as root. By default, root is not limited by quotas; therefore, the fileset quotas at home could be exceeded due to writes in cache.

The asynchronous commands are: **write**, **chmod**, **chown**, **create**, **mkdir**, **remove**, **rmdir**, **rename**, **link**, **symlink**, and attribute updates.

For more information, see *mmchfileset command* in *IBM Spectrum Scale: Administration and Programming Reference*.

Synchronous operations

Synchronous operations require an application request to block until the operation completes at the home cluster.

Read AFM does whole-file caching by default. In general, reading more than three blocks of the file drives AFM to cache the full file in the background for performance. To avoid reading large files when an application might be “peeking” into the file (for example, GUI viewers that read a few bytes to detect file mime type), certain file reads at the beginning of the file are handled individually. Sequential requests are pipelined between the application and gateway nodes to improve performance. This means data can be “streamed” from the home cluster to the application at the cache cluster.

Only files that have all blocks read, or the entire file contents fetched, are marked as cached. An uncached file cannot be evicted, resynched with home, or failed over to a new home. These AFM features are discussed in subsequent sections of this guide.

File and Directory Information

As users traverse the directory tree of the AFM-enabled fileset, files and directory attribute information from the home cluster is cached in GPFS on-demand. Henceforth, commands such as **ls** or **stat** continue to work even if the cache is subsequently disconnected from the home cluster. If file metadata information changes (via users at the home cluster or a different cache cluster), the cache is refreshed on demand to reflect new updates to existing files and creation of new files and directories. To reduce the number of operations sent to the home server, this refresh is done based on the validity lag configured. For more information, see *mmcrfileset command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Update synchronization

All modifications to the cached file system are sent back to the home cluster in the following situations.
- The end of the synchronization lag.
- If a synchronous command depends on the results of one or more updates; it synchronizes all depending commands to the server prior to its execution.
- An explicit flush of all pending updates using the **mmafmctl flushPending** command.

For more information, see *mmafmctrl command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Components of a cluster that is running AFM

This topic describes the communication, caching, and gateway behaviors on an IBM Spectrum Scale cluster that is running Active File Management (AFM).

Communication

Communication for caching between GPFS clusters is done by one or more gateway nodes. A node can be designated as a gateway node by using the **mmchnode** command. In addition to the gateway node designation on the cache cluster, the target path on the home server needs to be NFS-exported on any one node in the home cluster. This exported path of the home cluster is used in the cache cluster as a target path.

Active File anagement uses NFSv3 or GPFS protocol for communication between clusters. AFM also keeps track of server availability and disconnects or reconnects accordingly.

Caching behavior

Active file management is designed to be efficient on a WAN by minimizing the amount of traffic sent over the WAN. To improve efficiency, there are various parameters that allow controlling of caching behavior; for example, **afmFileOpenRefreshInterval** defines how often an open call checks for file changes at home.

When a file is being read into a cache file, data can be read as soon as it arrives in the cache. This means that the entire file does not have to be in cache to start reading it.

A cache is defined at the fileset level, so multiple cache filesets can share a single file system. This allows you to optimize the performance and space usage within existing storage, and it gives you flexibility in how you utilize cache relationships.

You can associate as many caches to a home as the bandwidth of the home cluster allows. The cache-to-home relationship is one-to-one, and the cache does all the work. You can think of AFM as a subscription service. The home is a data feed, and one or more caches can “subscribe” to that feed. This is what makes AFM very scalable; the GPFS cluster does not need to actively manage, as one data set, hundreds of relationships and the location on billions of files.

Gateway behavior

In a cluster with multiple gateway nodes and many AFM caches, AFM uses an improved hashing algorithm to elect the MDS for each of the caches. The improved hashing algorithm is set by default on a 4.1 cluster. On an upgraded cluster, the old hashing algorithm is effective. To move to the improved algorithm, the **mmchconfig afmHashVersion=2** command must be run after updating to the latest release. This command needs a fileset re-link or file system remount to take effect.

When node failures occur on the MDS of a cache, AFM elects a new MDS for the cache using the hashing algorithm and runs AFM recovery on the newly elected MDS. The new MDS thereafter takes over as owner for the cache. When an old MDS returns back after node failure, AFM transfers the queues as such from the current MDS to the old MDS. A change in MDS due to a node that has changed its gateway designation using **mmchnode** goes through AFM recovery.

For more information, see *mmchnode command* and *mmchnconfig command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Global namespace

This topic describes how Global namespace is implemented in IBM Spectrum Scale using Active File Management (AFM).

Home and cache entities can be combined to create a global namespace. Any client node in the world can use the same path to connect to the data within any of the GPFS clusters that are part of the namespace. In such a global namespace, the following AFM features can improve application performance at a remote site:

- When a file is being read into a cache, the data can be read as soon as it begins to arrive in the cache.
- Multiple cache filesets can share a single file system.
- Data can be transferred between sites in parallel.

When using GPFS multi-cluster as well to mount file systems between clusters, the benefit of using AFM is performance with high latency or unreliable networks.

Figure 12 shows an example of a global namespace, implemented using AFM, with three different sites. A GPFS client node from any site sees all of the data from all of the sites. Each site has a single file system. Each site is the home for two of the subdirectories and cache filesets pointing to the data originating at the other sites. Every node in all three clusters has direct access to the global namespace.

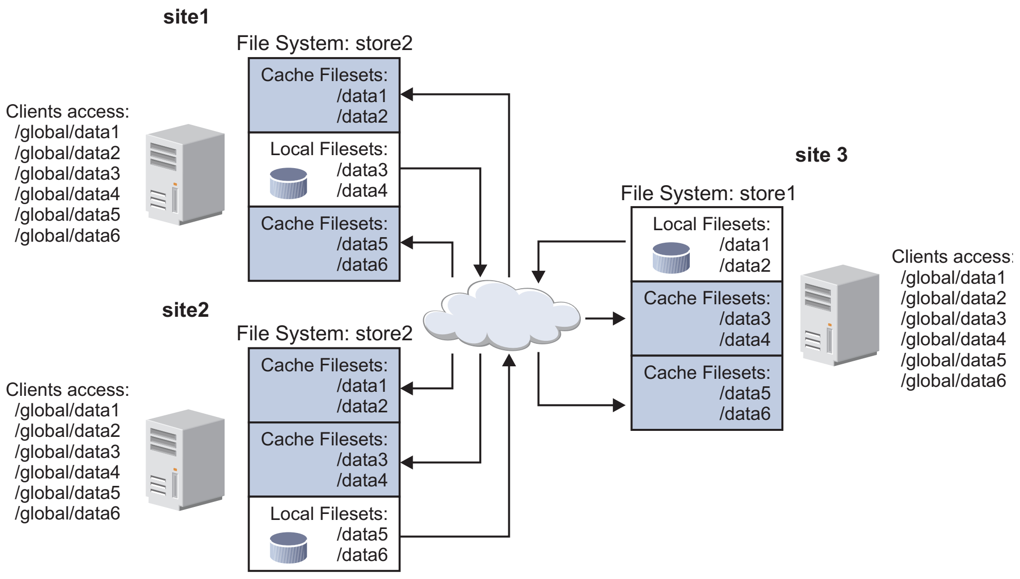

site1
File System: store2
Cache Filesets: /data1 /data2
Local Filesets: /data3 /data4
Cache Filesets: /data5 /data6
Clients access: /global/data1 /global/data2 /global/data3 /global/data4 /global/data5 /global/data6
site2
File System: store2
Cache Filesets: /data1 /data2
Cache Filesets: /data3 /data4
Local Filesets: /data5 /data6
Clients access: /global/data1 /global/data2 /global/data3 /global/data4 /global/data5 /global/data6
site 3
File System: store1
Local Filesets: /data1 /data2
Cache Filesets: /data3 /data4
Cache Filesets: /data5 /data6
Clients access: /global/data1 /global/data2 /global/data3 /global/data4 /global/data5 /global/data6

Figure 12. Global namespace implemented using AFM

Cache eviction

Cache eviction is enabled by default and is controlled by the **afmEnableAutoEviction** parameter and setting fileset block quota. Eviction is enabled by default.

Cache eviction can also be triggered manually using the **mmafmctl evict** command. There might be a time lag between the time that an eviction is triggered and the time that the data is actually evicted. Tuning the soft and hard quota limits minimizes application failure caused by data being cached at a faster rate than it is being evicted. Cache eviction for exceeding the inode limit is not supported.

See "Filesets and quotas" on page 57.

Disconnected operation

With the home cluster across a wide-area network, it is very likely that network connectivity experiences frequent intermittent outages and occasional long term disruptions. The reasons for these outages or disruptions can vary and can include faulty network hardware, routing or connection problems to the home cluster, NFS server failure at the home cluster, or even failure of the home cluster file system itself.

In the event that one or more gateway nodes determine that the home cluster cannot be accessed, all the caches that have a disconnected home go into disconnected mode. The behavior in a disconnected setting is slightly different for the two classes of operations. All synchronous operations are handled locally. Application requests for objects that are not yet in the cache return an error that the object does not exist (or an I/O error). On the other hand, all asynchronous operations can still update the cache and return successfully to the application. These requests remain queued at the gateway nodes pending reconnection. Updates to home experience a synchronization lag equal to the duration of the disruption in connectivity.

Note however, when the home is not reachable during revalidation, the cache waits until the **afmDisconnectTimeout** before it declares the outage. Writes requiring revalidation with home might appear temporarily stuck if it is done within the interval before the cache declares a home as disconnected. This is due to the revalidation being stuck on the unavailable NFS mount to respond and this would bail out after **afmDisconnectTimeout**.

Appending to or truncating an uncached file in a writer mode (SW or IW) sends only the updated contents to home on reconnect. Old contents at home are lost.

Note that because AFM caches file metadata separately from data, it is possible that one is able to see the attributes of a file using the **ls** or **stat** command, but not be able to fetch its contents from home in disconnected mode. In the case of caches based on native GPFS protocol, unavailability of the home file system on the cache cluster puts the caches into unmounted state. These caches never enter the disconnected state. For AFM filesets that use GPFS protocol to connect to the home cluster, if the remote mount becomes unresponsive due to issues at the home cluster not related to disconnection (such as a deadlock), operations that require remote mount access such as revalidation or reading un-cached contents also hang until remote mount becomes available again. One way to continue accessing all cached contents without disruption is to temporarily disable all the revalidation intervals until the home mount is accessible again.

It is also important to note that if the cache cluster is disconnected for an extended period, the number of file system updates might exceed the buffering capacity of the gateway nodes. In this situation, all changes continue to be "logged" in stable storage. Upon reconnection, AFM reads the changes from disk and synchronizes its local updates with the home cluster.

For more information, see *mmchfileset command* in *IBM Spectrum Scale: Administration and Programming Reference*.

Expiration

Read-only filesets can be configured to cause cached data to expire after the gateway nodes have been in a disconnected state for a specified amount of time.

This prevents access to stale data, where staleness is defined by the amount of time that the WAN cache is out of synchronization with data at the home site. Single-writer and local-update filesets cannot be configured to expire. The administrator can manually specify that the data in the cache is either expired or not expired using the **mmafmctl expire**/**unexpire** command.

Caches created using GPFS protocol do not go to expired state, even if expiration timeout is set for them. They continue to remain in unmounted state as long as the home file system is unavailable.

For more information, see *mmafmctrl command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Cache states

An AFM cache can have a different state depending on fileset and queue states. The **mmafmctl getstate** command displays the current cache state.

Possible cache states are the following:

Active This state indicates that the cache is active and ready for operations.

Dirty This state indicates that there are pending changes in cache that are not yet played at home. This state does not hamper user function, and the user can continue normal activity on the cache.

Disconnected
This state can occur only in a cache that is created over NFS export. It occurs when the MDS cannot connect to the NFS server at home. When parallel I/O is configured, this state shows the connectivity between the MDS and the mapped home server, irrespective of other gateway nodes. See "Parallel I/O" on page 227 for more details. To come out of this state to **Active** state, the administrator must correct the errant NFS server or servers on the home cluster.

Dropped
A cache fileset state moves to **Dropped** due to different reasons. Some reasons include:

1. Recovery failed

   The reasons for this failure can include the local file system being full, no space on the cache, or a policy failure during recovery. The administrator must rectify the issue and retry recovery or failback. The cache needs to be accessed to re-trigger recovery and to re-initiate failback, the **mmafmctl failback** command should be re-run.
2. Failback failed if fileset is IW

   The reasons for this failure can include the local file system being full, no space on the cache, or a policy failure during recovery. The administrator must rectify the issue and retry recovery or failback. The cache needs to be accessed to re-trigger recovery and to re-initiate failback, the **mmafmctl failback** command should be re-run.
3. A cache with active queue operations is forcibly unlinked. While all queued operations are being de-queued, the fileset remains in dropped state and moves to inactive state when the unlinking is complete. There is no administrative action required for this temporary dropped state.
4. While AFM internally performs queue transfers from one gateway to another to handle gateway node failures. This usually gets rectified automatically on the next accessing of the cache.
5. Export problems at home such as the following:

a. If `fsid` is not given before starting operations on the fileset on all NFS server nodes interacting with the cache clusters. Even if `fsid` provided later after operations have started on the fileset, things might not work.
b. Changing `fsid` on the home side after fileset operations have begun.
c. All home nfs servers do not have the same `fsid`for the same export path.

The following steps must be followed for exporting a new path on home side for AFM filesets:

1. Ensure that the FSID value is same for given share on all home side nodes in `/etc/exports`.
2. If an `fsid` 'f' given for child does not provide for parent.

   If `fsid` 'f' is provided `/gpfs1/fset1` and `/gpfs1` is also being exported, do not give same `fsid` for `/gpfs1`.
3. Do "nfs restart" or **exportfs -a** on all home side nodes.
4. Run **mmafmconfig enable** *ExportPath* on home file system.
5. After this only, home export is ready to use by AFM cache side and user can Create fresh filesets.

If new exports are defined after cache filesets are created, the cache filesets can be re-linked to move them to **Active** state.

Expired
When an RO cache is disconnected, the cached contents are still accessible for the user. However, the administrator can define a time from home beyond which access to the cached contents becomes stale. Such an event that occurs automatically after disconnection (when cached contents are no longer accessible) is called expiration; the cache is said to be expired. This state is not applicable to any other mode except RO.

FailbackCompleted
This state is applicable only for IW cache. This state occurs when a failback has successfully completed on an IW cache fileset. The administrator must run **mmafmctl failback --stop** to move the IW cache to **Active** state.

FailbackInProgress
This state is applicable only for IW cache. It occurs when a failback process has been initiated on an IW cache fileset and it is in progress. This moves automatically to **FailbackCompleted** once the failback process has completed.

FailoverInProgress
This state shows that the cache is in the middle of a failover process. This moves to **Active** when the failover is complete.

FlushOnly
This state indicates that operations are queued but have not started to flush. This is a temporary state and should move to **Active** when a write is initiated.

Inactive
This state is possible when a fileset has been just created and operations have not been initiated on the fileset. This should move to **Active** once operations begin.

NeedsFailback
This state is applicable only for an IW cache. It can occur when a failback initiated on an IW cache is interrupted and is incomplete. Failback automatically gets triggered on the fileset, or the administrator can rerun failback.

NeedsResync
This state is applicable for SW cache, when the cache detects some accidental corruption of data at home. The **mmafmctl resync** command needs to be run on the fileset to move it to **Active** state.

QueueOnly

This state is applicable for SW/IW caches. A cache fileset is moved to **QueueOnly** when operations at the cache are queued but not yet flushed. This can happen in states such as recovery, resync, failover when the queue is in the process of getting flushed to home. This state is temporary and the user can continue normal activity.

Recovery

This state is applicable only for SW/IW caches. A cache is said to be in recovery state when it is running the recovery process and is in the process of queueing or flushing the updates. This state does not hamper normal user function, and the state moves to **Active** once recovery is complete.

Unmounted

Caches using NFS as transport protocol go into **Unmounted** state if home NFS is not accessible, or if home exports are not exported properly or home export does not exist. The problem with home exports must be resolved. After 300 seconds, the cache retries connecting with home and moves to **Active** state.

Caches using native GPFS protocol as target move to **Unmounted** state when there are problems accessing the local mount of the remote file system. To resolve this problem, the remote file system needs to be remounted on the local cache cluster.

A cache fileset can also move to **Unmounted** state when the cache detects a change in home, such as accidental deletion of the exported independent fileset at home.

Failure and recovery

Failures can occur in AFM if either an application node or a gateway node fails. Failures are not catastrophic and do not result in the loss of data or the loss of AFM's ability to update the home cluster with local modifications.

AFM stores, on disk, all necessary state to “replay” updates made to the cache at the home cluster in case the in-memory queue at the gateway node is lost due to a node failure or memory pressures. The recovery process recovers any object that does not belong to a snapshot. Psnaps lost in queue due to gateway node failures cannot be recovered through recovery.

The **mmafmconfig** command must have been run at home for using the AFM recovery feature. Since this is not available on non-GPFS file systems at home, recovery does not work with non-GPFS home.

Application node failure

Since application nodes are just regular GPFS nodes that can also access AFM filesets, failure of one of them is handled just like a regular node failure. No special handling is required for AFM.

Gateway node failure

If there were no updates made to any AFM-enabled fileset, then the failure of a gateway node is harmless and application nodes do not experience requests delays. While in recovery mode, application requests to all AFM-enabled filesets are momentarily blocked.

On the other hand, if pending updates for any AFM-enabled fileset exist, for example, file write or create, then the failure of a gateway node puts the entire cache cluster into recovery mode.

In recovery mode, the cache examines the requests in cache that have not been sent to home and queues all these pending operations on the gateway nodes. AFM collects the pending operations by internally running a policy scan on the files in the fileset. AFM leverages the policy infrastructure in GPFS to use all the nodes mounting the file system. Therefore, the availability of all these nodes mounting the file system is required for recovery to complete. When this queue is completely played at home, according to the synchronization lag, the cache and home become synchronized in contents and recovery is said to be completed. The cache goes back to **Active** state.

Recovery is triggered only for single-writer mode or independent-writer mode. Recovery can run in parallel on multiple filesets, although only one instance can run on a fileset at one time.

Handling an IW cache fileset disaster

In cases where an IW cache is down, either permanently due to disaster or temporarily due to maintenance, the applications can be moved to the home cluster.

The nature of AFM is to push the data asynchronously, so the cache might have pending updates that have not been pushed to home at the time when the applications are moved. In cases where application semantics allow them to move from cache to home with pending updates still in cache, it is possible that files at home will change along with the pending data in cache.

Once the cache comes back, the first access drives the cache to push stale data to home. This might result in a stale pending update to overwrite the latest data written at home while the cache was down. In these cases, the following steps must be followed so that latest data is preserved and old or stale data is discarded. The data staleness is determined based on file mtime and failover time (the time when applications were moved to home, given as input to failback command).

In the case of an unplanned DR, follow these administrative actions:

1. When the cache site is ready to take over again, stop applications at the home site.
2. Link the cache site back or bring the gateways up. Do not access the fileset directories.
3. Run the following command from the cache site to sync up the latest data (specifying the time when the home site became functional and including the complete time zone, in case home and cache are in different time zones):

   ```
   mmafmctl FileSystem failback -j Fileset --start --failover-time 'TimeIncludingTimezone'
   ```

   The failback command resolves any conflicts between pending updates in cache at the time of failover with the data changes at home. Any new files created at home are fetched on next access to those respective files or directories based on the revalidation interval. In cases where an application needs to know these new files upfront, they can be fetched via **mmafmctl prefetch** before starting applications on cache. When the sync-up is in progress, the fileset state is **FailbackInProgress**.

   Note: If failback does not complete or if the fileset moves to **NeedFailback** state, rerun the failback command.

   After failback completes, the fileset state moves to **FailbackCompleted**.
4. After achieving the **FailbackCompleted** state, run this variation of the **mmafmctl failback** command to move the fileset to **Active** state:

   ```
   mmafmctl FileSystem failback -j Fileset --stop
   ```

   The cache site is ready for use. All applications can start functioning from the cache site.

   This option moves the cache from **Read-Only** state to **Active** state, where it is ready for use. Failback does not pull in uncached data from home, which needs to be done explicitly by the administrator using the **mmafmctl prefetch** utility. This can be done before running the stop option to ensure that cache has all the latest data from home before new changes are made.

As a result of an unplanned outage, some updates from the cache site might be lost. These lost updates are handled when the cache site syncs up with the latest data updates from the home site.

The cache identifies any dirty data that did not get synced up and does the following:

1. If the file is not modified at the home when the applications were connected to the home, the cache pushes the change to the home so that there is no loss of data.
2. If the file was most recently modified at the home, the cache discards the earlier updates from the cache. The next lookup on the cache updates the latest metadata from the home.

If the conflicted dirty data is a directory, it is moved to the `.ptrash` directory. If the conflicted dirty data is a file, it is moved to the `.pconflict` directory. The administrator needs to clean the `.pconflict` directory from time to time, as it is not cleaned automatically. When IW is converted to other modes, the `.ptrash` and `.pconflict` directories remain.

In RO or LU mode when there is a delete conflict with a file (that is, a file is removed at home but is still existing in cache), the conflict is resolved by removing the file from the cache. In SW or IW mode, if such a file is non-dirty, it is moved to `.ptrash`, and if it is dirty, it is moved to `.pconflicts`.

Files moved back from `.ptrash` or `.pconflicts` are treated as local files to the cache fileset and are not queued to home. However, if they are copied to the cache, they are queued to home as new files.

The fileset behaves as a read-only fileset while failback is in progress.

Creating hard links at home or in cache is not recommended when IW is used.

If failback is expected to be used, the home file system must update the ctime of the object on a RENAME operation. This is enabled using **setCtimeOnFileRename** at home.

This can be done using the **mmchconfig** command at home:

```
mmchconfig setCtimeOnFileRename=yes -i
```

For additional information, see the topic *Failover of IBM Spectrum Scale cache filesets*.

Prefetching

Wide Area Network (WAN) caching fetches complete files on demand from the home fileset to the remote system cache in real time during normal operation. The prefetching feature moves files into the cache in batch mode so that they are already present in the cache when they are accessed by an application. The prefetching of cache before an application is started can reduce the network delay when the application starts.

Prefetching can also be used to proactively manage WAN traffic patterns by moving files over the WAN during a period of lower WAN usage in anticipation that it might otherwise be accessed during a period of higher WAN usage.

GPFS can prefetch the data using the **mmafmctl** *Device* **prefetch –j** *FilesetName* command (which specifies a list of files to prefetch). Note the following about prefetching:

- It can be run in parallel on multiple filesets (although more than one prefetching job cannot be run in parallel on a single fileset).
- If a file is in the process of getting prefetched, it is not evicted.
- If the MDS of a cache is changed while prefetch is running, prefetch gets interrupted and does not reach completion. The next access to the fileset automatically re-triggers the interrupted prefetch on the new MDS. For re-triggering prefetch after gateway node failures, the list file used when prefetch was initiated must exist in a path that is accessible to all gateway nodes. If parallel I/O is configured, all gateways participate in the prefetch process.

Peer snapshots

The cache fileset peer snapshot function provides a generic infrastructure for obtaining an application-consistent point-in-time copy of data in the cache fileset that can be integrated with other functions to meet specific customer requirements such as backup and recovery.

The peer snapshot pushes all of the snapshot data in cache to the home fileset so that the home data is consistent with cache, then takes a snapshot of the corresponding home data. This results in a pair of peer snapshots, one each at the cache and home filesets, that refer to the same consistent copy.

Note: If cache is disconnected from the home fileset when the cache snapshot is created, the cache notes that the peer snapshot on the home fileset has not been created. When the home fileset becomes reconnected to cache, the cache attempts to resume the creation of the missing peer snapshot at the home fileset if conditions permit. The **mmpsnap** command can be used to create a peer snapshot.

The last successful snapshot is saved and can be viewed via **mmlssnapshot**. Multiple outstanding peer snapshots can be queued on the gateway node. Peer snapshots must not be deleted via the normal **mmdelsnapshot** command but rather through the **mmpsnap** command to ensure that both the cache and home snapshots are removed. The **mmpsnap** command works only if the home cluster has executed the **mmafmconfig enable** *ExportPath* command. **mmpsnap** can be used only in an SW cache; it cannot be used for RO, IW, or LU caches.

It is not possible to create a peer snapshot when a cache is created using the GPFS protocol.

For more information, see *mmpsnap command* in *IBM Spectrum Scale: Administration and Programming Reference*.

Viewing snapshots at home

All snapshots at home for RO and LU filesets can be viewed in the cache by setting the **afmShowHomeSnapshot** parameter to **yes**. This variable is not applicable for SW and IW filesets.

The home and cache can have different snapshot directories for clarity. If the **afmShowHomeSnapshot** value is changed after fileset creation, the change is reflected in cache only after the next GPFS recycle or file system remount. However, if the value is changed from **yes** to **no** at any time during the lifetime of a fileset, it continues to show the home snapshots even after GPFS recycle or file system remount.

Also, for RO and LU filesets using NFS backend, the changes at home to the snapshot directory do not get reflected in cache. This is because NFS cannot detect the changes on `.snapshot` directory as the mtime/ctime of this directory does not get modified at home despite snapshots creates or deletes at home. This is not true for RO and LU filesets using GPFS backend as the latest snapshot directory gets reflected in cache.

Failover of cache filesets

AFM filesets continue to function independent of the home, in the event of home failures. The AFM filesets serve the cache applications with cached data. When a new home replaces the failed home, the administrator must run the **mmafmctl** *Device* **failover -j** *Fileset* command to let the cache point to the new home. The home is expected to be empty when this command is run.

If failover is interrupted (for example, due to gateway node failure or quorum loss), the next access that requires access to home will restart it automatically.

Caches in SW and IW can be failed over to a new home. Caches in RO and LU cannot be failed over to a new home. Failover fills the new home with the contents from the cache. Both cached data and metadata from cache are pushed to the new home. To ensure that extended attributes get synced, the new home must have run the **mmafmconfig** command before the failover command.

When there are multiple IW caches, the administrator must choose a primary IW cache and fail this over to a new empty home. It is recommended that all the other IW caches to the old home be deleted and recreated. If another of the IW caches is failed over to the home after failing over the primary IW cache, all the stale data in that cache overwrites the objects of the same type and name at home. For this reason, failover to a non-empty home is discouraged.

Failover does not work if the new target is an NFS mapping. However, this restriction does not apply if target is GPFS.

Failover does not support map name as part of the target.

There are no implications for failover and resync when applied to a GPFS target. It is possible to change the protocol along with the target using failover. For example, a cache using an NFS target `bear110:/gpfs/gpfsA/home` can be switched to a GPFS target whose remote file system is mounted at `/gpfs/fs1`, and vice-versa as follows: failover does not support map name as part of the target.

```
mmafmctl fs0 failover -j afm-mc1 --new-target gpfs:///gpfs/fs1
mmafmctl fs0 failover -j afm-mc1 --new-target nfs://bear110/gpfs/gpfsA/home
```

Note: Each AFM fileset is independently managed and has a one-to-one relationship with a target. This allows different protocol backends to coexist on separate filesets in the same file system. However, AFM does not validate the target for correctness when the fileset is created. It is the responsibility of the user to specify a valid target; using a GPFS target that belongs to the same file system as the AFM fileset might lead to undesirable consequences. Also, note that admin must disable automation eviction when home fails and before starting failover. This is to make sure that eviction does not free cached data on the cache. There is no way to recover evicted data in case home cluster fails.

mmafmctl failover with **--target-only** option can be used if the user wants to change the mount path/IP address in the target path. The new NFS server should be in the same home cluster and should be of the same architecture as the existing NFS server in the target path. This option must not be used to change the target location or protocol.

Partial file caching

Partial file caching fetches only blocks that are accessed, thereby utilizing network and local disk space more efficiently.

Partial file caching is useful if an application does not need to read the whole file. Partial file caching is enabled on a GPFS block boundary.

For sparse files, the percentage for prefetching is calculated as the ratio of the size of data blocks allocated in cache and the total size of data blocks at home. Holes at the home file are not considered in the calculation.

Partial file caching is controlled by the **afmPrefetchThreshold** configuration attribute. For details, see the **mmchfileset** command description in the *IBM Spectrum Scale: Administration and Programming Reference*. .

Prefetch of partially cached files pulls the complete file to the cache. Failover of partially cached files pushes only the cached data blocks to the new home. Uncached blocks are filled with null bytes.

Any writes queued on a partially cached file will pull in the whole file, even if a prefetch threshold is set on the file.

AFM encryption support

AFM supports file encryption. In some cases, communication between the clusters should be encrypted explicitly by configuring a recommended cipher.

Encryption can be applied to AFM-managed filesets. AFM home sites, cache sites, or both can be enabled with encryption, independently. The data is encrypted while at rest (on disk) and is decrypted on the way to the reader/application hosted on home and caches; however, AFM communication between home and cache is not encrypted.

Because the data that flows between home and caches is not encrypted by the adoption of file encryption, communication between the clusters needs to be encrypted explicitly (if privacy of the data over the network is a concern), by ensuring that a cipher list is configured. To ensure that data is transmitted in encrypted form, a cipher other than **AUTHONLY** should be adopted. **AES128-GCM-SHA256** is one of the recommended ciphers. Issue command **mmauth show** to display the cipher lists used to communicate within the local and with the remote clusters. To ensure that all file content at rest (on disk) and on the network are encrypted, configure file encryption at home and on the caches, and also configure a cipher list on all the clusters, ensuring ciphers are configured within and across clusters. Even though file encryption results in data that is transmitted in encrypted form between NSD clients and servers (both directions), neither file metadata or RPC headers are encrypted. Only the use of encrypted communications (cipher list) ensures that the entire message content gets encrypted.

If the NFS protocol is being used for communication between the home and cache clusters, and privacy of the data over the network is a concern, then an encrypted tunnel needs to be set up to encrypt the NFS transfers. If a cluster configured as cache includes an encrypted file system, then all nodes in the cluster, and especially the gateway nodes, require access to the Master Encryption Keys.

For encryption setup, see Chapter 19, "Encryption," on page 367.

Parallel I/O

The gateway acting as the MDS of a cache fileset is the channel for communication with the home cluster. To help the MDS exchange data with the home cluster for large-size I/Os, a cache cluster can be configured to leverage all nodes in the cluster configured as gateways.

For this function, multiple NFS servers are required at the home cluster. In a cache cluster that uses NFS as transport protocol, each gateway node in the cache cluster can be mapped to a specific NFS server at home. This mapping aids I/O load distribution.

One or more gateway nodes can be mapped to the same NFS server, but one gateway node cannot be mapped to more than one NFS server. Mapping defined using **mmafmconfig** needs a GPFS recycle to take effect on existing filesets. If there is no mapping specified or if a matching failed, parallelism is not effective.

On an AFM cluster using the native GPFS protocol for defining an AFM target, gateway nodes can be mapped to any other node in the same cache cluster. In the absence of a mapping definition, all gateway nodes are used for the I/O.

The following example shows a mapping for NFS target, assuming cache gateway servers `hs22n18`, `hs22n19`, `hs22n20`, and `hs22n21`, and home NFS servers `js22n01` and `js22n02`:

```
# mmafmconfig add js22n01 --export-map js22n01/hs22n18,js22n02/hs22n19
mmafmconfig: Command successfully completed
mmafmconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

```
# mmafmconfig add js22n02 --export-map js22n02/hs22n20,js22n01/hs22n21
mmafmconfig: Command successfully completed
mmafmconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The **mmafmconfig** command can be used to display, delete, or update mappings. New changes in the active mapping between gateway nodes and home NFS servers takes effect only after fileset re-link or file system remount. Gateway designation can only be removed from a node if the node is not participating in an active mapping.

```
# mmafmconfig show
Map name:            js22n01
Export server map:   192.168.200.12/hs22n19.gpfs.net,192.168.200.11/hs22n18.gpfs.net

Map name:            js22n02
Export server map:   192.168.200.11/hs22n20.gpfs.net,192.168.200.12/hs22n21.gpfs.net
```

Parallel reads and writes need to be configured separately. Parallel reads or writes are effective for files whose sizes are larger than those specified by the respective parallel threshold. The threshold can be defined using **afmParallelWriteThreshold** and **afmParallelReadThreshold** parameters. This holds good for reads and writes on all types of files except reads on sparse files and reads on files with partial file caching enabled, which is served only by the MDS without splitting.

The MDS of a cache fileset communicates with the each of the participating gateway nodes based on their availability. A single I/O request is split into multiple chunks. These chunks are sent across all gateway nodes in parallel. The size of each chunk is configurable using the **afmParallelWriteChunkSize** and **afmParallelReadChunkSize** parameters.

The gateway nodes communicate with their respective NFS servers and get a reply. If more than one gateway node is mapped to the same NFS server at home, only one among the gateway nodes is chosen to perform a read task. However, a write task is split among all the gateway nodes. This is not applicable to caches created using native GPFS protocol, which use the mapping specified or all gateway nodes wherever no mapping is specified.

For Parallel I/O to work, it is recommended that all the NFS servers being used for the mapping at cache to have the same architecture. For example, if the NFS server that the MDS is of System x architecture and a participating GW is of Power Architecture®, such a participating GW is not useful for Parallel I/O.

The MDS processes the replies from all gateway nodes. The MDS handles all I/O failures from the participating gateway nodes and coordinates all the activities until the IO is completed. In case of any failure on any participating gateway node, MDS attempts to retry the failed task on next available participating gateway and logs an error message into `mmfs` logs. If all participating gateway nodes fail, the MDS retries the operation.

Configuration parameters for parallel reads and writes are discussed in detail in "Tuning active file management home communications" on page 232.

All AFM functions such as recovery, resync, prefetch except failover, use Parallel I/O when configured.

Disabling AFM

An AFM fileset can be converted to a normal independent fileset. An independent fileset cannot be converted to an AFM fileset.

In scenarios where all data from home is destroyed or is not available, the AFM fileset can be converted to an independent fileset by using the **mmchfileset -p afmTarget=disable** option after unlinking the fileset. Thereafter, the fileset behaves like a regular GPFS independent fileset and no requests are queued to home.

Disabling targets for AFM filesets is intended only for RO or LU filesets, not for writer filesets. This feature is useful in NFS migration use cases, where AFM can be disabled after migration. Migration is discussed in detailed in NFS migration section.

For details on migrations, see "AFM-based NFS migration."

AFM-based NFS migration

NFS migration is a process of migrating data from any legacy storage appliance to a GPFS cluster via NFS protocol.

This is useful in the event of upgrading hardware or buying a new system where the data from old hardware needs to be moved to new hardware. Minimizing application downtime and moving data with attributes are key goals of migration.

The target or new hardware should be running GPFS 4.1 or later. The data source should be an NFS v3 export and can be either a GPFS or a non-GPFS source. Any source running a version earlier than GPFS 3.4 is equivalent to a non-GPFS data source. The UID namespace between source and target must be maintained.

Migration can be incremental or progressive, based on how it is performed. The old storage appliance can be disconnected once the migration is complete.

Migration does not pull file system–specific parameters such as quotas, snapshots, file system–level tuning parameters, policies, fileset definitions, encryption keys, and dmapi parameters. On a GPFS data source, AFM moves all user extended attributes and ACLs, and file sparseness is maintained. On a non-GPFS data source POSIX permissions or ACLs are migrated, but not the NFS V4/CIFS ACLs, non-posix file system attributes, and sparseness.

AFM migrates data as root, bypassing permission checks. Pre-allocated files at the home source are not maintained; that is, only actual data blocks are migrated based on file size. If all objects in the fileset are not to be migrated and/or the fileset to be migrated has orphans, admin should clean these inodes before starting migration.

Prefetch across gateway node failures will be handled by AFM to continue after gateway nodes come back up. Refer to "Prefetching" on page 224 for more details.

Migration methods

The following migration methods are available:

Incremental migration

In this method of migration, the metadata tree and critical data required for running applications are populated in the GPFS AFM cluster. While this is completing, the applications continue to run in the source cluster. When the GPFS AFM cluster is pre-populated as close as possible, applications take a downtime and are moved to this cluster.

Detailed instructions follow:

1. The NFS shares whose data needs to be migrated should be identified on the source cluster or legacy NAS storage. All identified shares should be exported via NFS.
2. If the source is running GPFS 4.1 or later, run **mmafmconfig enable** on each of the NFS shares. If the version is GPFS 3.5 or 3.4, run **mmafmhomeconfig enable**.
3. An AFM RO fileset should be created at the GPFS AFM cluster for each NFS share. **afmAutoEviction** should be disabled for all the RO filesets, to inadvertently avoid eviction. It is advisable to preallocate inodes equal to or greater than the inodes present at the source,

as that many files are known to exist. The maximum number of inodes should also be set accordingly. Parallel I/O can be configured between the source NFS servers and the target. See "Parallel I/O" on page 227.

4. For each of the RO filesets, a list of all the files at the source should be created. If the source is GPFS, a simple policy LIST RULE can be used to create such a list. This policy can be tuned to eliminate types that are not supported by AFM. If the source is not GPFS, such a list can be created using **find**, **ls**, **-R**, or any similar tool. It must then be edited by hand to eliminate unsupported types.
5. The **mmafmctl prefetch** command with the **--metadata-only** option should be run on each of the RO filesets to populate the directory tree using the list file generated for the respective filesets.
6. Run prefetch for each RO fileset using the respective list files created previously. Prefetch has to complete on all filesets before proceeding.

   Callbacks can be added to indicate when a scheduled prefetch task is completed using the **afmPrepopEnd** event.(For details, see the **mmaddcallback** command description in the *IBM Spectrum Scale: Administration and Programming Reference* .) Once a prefetch task completes, migration of the given data is said to be complete.
7. Stop applications at the data source.
8. In order to sync up the latest application changes, steps 4 through 6 should be repeated. The **mmafmctl prefetch –metadata-only** command will sync up any new files or directories created by the application after the previous migration.
9. AFM filesets can be converted to SW or IW filesets with a new target, or to regular GPFS filesets with caching disabled.
10. Migration is complete, and applications can be started on the target system. The old system is ready to be decommissioned.

    It is recommended to maintain continuous connectivity between two systems when prefetch is running, to avoid prefetch failures. Prefetch tasks that fail due to disconnection need to be rerun.

Progressive migration

This is largely similar to incremental migration. The specific steps follow:

1. The NFS shares whose data needs to be migrated should be identified on the source cluster or Legacy NAS Storage. All identified shares should be exported via NFS.
2. If source is running GPFS 4.1 or later, run **mmafmconfig enable** on each of the NFS shares. If the version is GPFS 3.5 or 3.4, run **mmafmhomeconfig enable**.
3. An AFM LU fileset should be created at the GPFS AFM cluster for each NFS share. **afmAutoEviction** should be disabled for all the LU filesets, to inadvertently avoid eviction. It is advisable to preallocate inodes equal to or greater than the inodes present at source, as that many files are known to exist. The maximum number of inodes should also be set accordingly. Parallel I/O can be configured between the source NFS servers and the target.
4. Applications should be shut down at the source.
5. For each of the LU filesets, a list of all the files at the source should be created. If the source is GPFS, a simple policy LIST RULE can be used to create such a list. This policy can be tuned to eliminate types that are not supported by AFM. If the source is not GPFS, such a list can be created using **find**, **ls**, **-R**, or any similar tool. It must then be edited by hand to eliminate unsupported types.
6. The **mmafmctl prefetch** command with **--metadata-only** option should be run on each of the LU filesets to populate the directory tree using the respective list file created in the preceding step.
7. Applications are started at the target GPFS cluster.
8. While applications are running at the target, the **mmafmctl prefetch** command can be issued to run on each of the LU filesets, using the respective list files created in step 5. Prefetch

must complete on all filesets before proceeding. Callbacks can be added to indicate when a scheduled prefetch task is completed using the **afmPrepopEnd** event. (For details, see the **mmaddcallback** command description in the *IBM Spectrum Scale: Administration and Programming Reference* .) Once a prefetch task completes, migration of the given data is said to be complete.

9. Migration is complete when prefetch is successful on all the filesets. The old system is ready to be decommissioned.
10. The GPFS AFM filesets can be converted to regular GPFS filesets with caching disabled or can continue as LU filesets. In case the filesets continue in LU mode, it is recommended to disable the refresh intervals (File Lookup Refresh Interval, File Open Refresh Interval, Dir Lookup Refresh Interval and Dir Open Refresh Interval).

    The downtime required is only for moving applications to a new GPFS AFM cluster and for AFM fileset conversions at the end of migration. Applications are expected to experience some latency during the on-demand pull of data from the source.

Using HSM

If data is managed by HSM at the data source, migration will drive data recall from the tape onto the source and then to the target system.

For more information about AFM restrictions, see IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

HSM support on AFM filesets

This topic describes the behavior of HSM on AFM filesets.

When a file from a cache fileset reads a migrated file at home, the file at home is recalled and fetched at cache. If the cache file is partially cached and **afmPrefetchThreshold** is enabled, then the default behavior is to allow the partially migrated files at cache to be migrated to HSM servers. You can prevent this migration by setting `AFMSKIPUNCACHEDFILES yes` in the `dsm.sys` configuration file. For more information, see “Partial file caching” on page 226.

In cache filesets with the ability to write, if a dirty file is migrated, then flushing the queue recalls the file and makes it premigrated. You can prevent this migration by setting `AFMSKIPUNCACHEDFILES yes` in the `dsm.sys` configuration file.

Home cluster errors

This topic explains the handling of synchronous command and asynchronous update errors on the home cluster.

When a synchronous command experiences an error (such as reading a cached file while disconnected, or permission errors with the home cluster), it immediately reports the error back to the user.

If an asynchronous update (such as a create, write, or attribute change) fails due to any reason (for example, **EPERM**, **EROFS**, **EINVAL**, or configuration errors), an error message is written to the log, and the operation is left on the queue and put into a special state. All further operations that depend on the failed operation do not execute. These errors are logged in the `/var/adm/ras/mmfs.log.latest` file on the MDS of a fileset or a fileset gateway.

When this happens, you can look at the error in the log file and try to fix the problem. Once the problem is fixed, you can execute the **mmafmctl resumeRequeued** command on the gateway node with the failed message. This puts the operation back in a normal state and tries to execute it to the home cluster. If another error occurs, or the problem is still not fixed, the whole process starts over again.

If the process of fixing the error in the cache involves a GPFS restart or gateway reboot, then the queue might get dropped. This puts the cache out of sync with home, as discussed in “Failure and recovery” on page 222. The gateway goes into recovery mode when it comes up again.

If an abnormal error such as **E_STALE** is received from home for a write operation in IW mode, the cache causes the write to be dropped, with no recourse. An AFM error is logged in this situation, and the file is pulled in from home on the next revalidation. No corrective action needs to be taken by the administrator for these errors.

Administrative actions

This section describes the administrative actions associated with AFM.

Dealing with requeued messages

When there is an error in executing a message, AFM re-queues the message.

AFM tries to execute the message at a later point in time, and this continues in a loop until a message gets executed. In some cases, the user or administrator can correct the reason for the error and then force the message to be executed immediately, without waiting for AFM to execute in the next cycle. The **mmafmctl resumeRequeued** command can be used to execute the requeued messages in the queue.

For more information, see *mmafmctrl command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Handling inadvertent changes at home for SW

This topic explains how to reverse inadvertent changes made at the home for an AFM SW fileset.

An SW home is not writable. In case of inadvertent changes made at home of an SW fileset such as delete of a file or change of data in a file etc., you can requeue all contents from an SW cache to the home. This is done using **mmafmctl** command **resync** option. This does not fix renames done at home.

For more information, see *mmafmctrl command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

Tuning active file management home communications

This topic describes performance tuning on active file management home communications.

Note: Values suggested here reflect evaluations made at the time this documentation was written. For most recent tuning recommendations, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) and the IBM Spectrum Scale Wiki (www.ibm.com/developerworks/community/wikis/home/wiki/General Parallel File System (GPFS)).

AFM communication is done using the NFSv3 protocol; for peak performance it is a good idea to tune the gateway NFS servers that are hosting home exports and the gateway servers supporting cache filesets.

Most of these tuning parameters require at least the AFM client to be restarted. It is always safest to make sure that the NFS server is not mounted. You can achieve this by unlinking the AFM filesets or stopping GPFS on the gateway node.

There are significant differences between tuning for 1GigE and 10GigE networks. For 10GbE everything needs to be scaled up, but not necessarily by a factor of 10. Many of these settings do not survive reboots so they need to be set each time the server is started. The TCP buffer tuning is required for 1GigE links with rtt > 0 and for all 10GigE links (possibly even in a LAN).

Tuning on gateway nodes of the cache cluster (NFS Client)

The **sunrpc.tcp_slot_table_entries** or **/proc/sys/sunrpc/tcp_slot_table_entries** parameter sets the maximum number of (TCP) RPC requests that can be in flight. Leave as the default for 1GigE with no round-trip time. You might want to increase it beyond 16 if the round-trip time is large. For 10GigE, make sure this value is at least 48; possibly even higher depending on the round-trip time.

Tuning required on both the NFS client and the NFS server

You must set TCP values that are appropriate for the delay (buffer size = bandwidth * RTT).

For example, if your ping time is 50 ms, and the end-to-end network consists of all 100BT Ethernet and OC3 (155 Mbps), the TCP buffers should be the following:

```
0.05 sec * 10 MB/sec = 500 KB
```

If you are connected via a T1 line (1 Mbps) or less, the default buffers are fine; but faster networks usually benefit from buffer tuning.

The following parameters can also be used for tuning (note that a 12194304 buffer size is provided here as an example value for a 1GigE link with a delay of 120ms). To set these values, place the following settings in a file, then load it with **sysctl -p** *filename*.

The following are example values; initial testing may be required to determine the best values for a particular system:

```
net.ipv4.tcp_rmem = 12194304 12194304 12194304
net.ipv4.tcp_wmem = 12194304 12194304 12194304
net.ipv4.tcp_mem = 16777216 16777216 16777216
net.core.rmem_max = 12194304
net.core.wmem_max =  12194304
net.core.rmem_default = 12194304
net.core.wmem_default = 12194304
net.core.optmem_max = 12194304
net.core.netdev_max_backlog = 250000
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 1
```

Note: For TCP tuning, the **sysctl** value changes do not take effect until a new TCP connection is created (which occurs at NFS mount time). Therefore, for TCP changes, it is critical that the AFM fileset (and thus the NFS client) is unmounted and GPFS is shut down.

With Red Hat Enterprise Linux 6.1 and later, both the NFS client and server perform TCP autotuning. This means that it automatically increases the size of the TCP buffer within the specified limits (via **sysctl**). If either the client or server TCP limits are too small, the TCP buffer grows as needed for various round-trip time between GPFS clusters. With versions earlier than Red Hat Enterprise Linux 6.1, NFS is limited in its ability to tune the TCP connection. Therefore, do not use a version earlier than Red Hat Enterprise Linux 6.1 in the cache cluster.

Since a GPFS cluster might be handling local and remote NFS clients, one strategy is to set the GPFS server values at the value for the largest expected round-trip time of any NFS client. This ensures that the GPFS server can handle clients in various locations. Then on the NFS clients, set the TCP buffer values that are appropriate for the SONAS cluster that they are accessing.

For AFM, the gateway node is both an NFS server (for standard NFS clients if they exist) and an NFS client (for communication with the home cluster). Ensure that the TCP values are set appropriately, as values that are either too big or too small can negatively impact performance.

If performance continues to be an issue, try increasing the buffer value by *at most* 50%. Any further increase in size will have a negative effect on performance.

NFS server tuning on the home cluster

/proc/fs/nfsd/max_block_size
Set this to 1MB for improved performance.

/proc/fs/nfsd/threads
Set to a minimum value of 32, but possibly greater than 128 depending on the throughput capacity and round-trip time between the cache and home clusters. You might need to experiment to determine the correct value.

nfsPrefetchStrategy
Since AFM uses NFS, ensuring that this is set on the home GPFS cluster is critical. Set it to at least 5 to 10.

Tuning active file management

The following table lists AFM configuration parameters with their default values and the commands used to set them initially or change them.

Table 43. Active file management configuration parameters and default values

AFM configuration parameter	Valid values	Default value	Tunable at cluster level	Tunable at fileset level	Notes
afmAsyncDelay	1 through 2147483647	15	Yes	Yes	This is an optional attribute that is set by default when the fileset is created.
afmDirLookupRefreshInterval	0 through 2147483647	60	Yes	Yes	This is an optional attribute that is set by default when the fileset is created.
afmDirOpenRefreshInterval	0 through 2147483647	60	Yes	Yes	This is an optional attribute that is set by default when the fileset is created.
afmDisconnectTimeout	0 through 2147483647, **disable**	60	Yes	Yes	This is an optional attribute that is set by default when the fileset is created. Only **mmchconfig** can be used to change this attribute.
afmExpirationTimeout	0 through 2147483647, **disable**	300	Yes	Yes	This is an optional attribute that is set by default when the fileset is created.

Table 43. Active file management configuration parameters and default values (continued)

AFM configuration parameter	Valid values	Default value	Tunable at cluster level	Tunable at fileset level	Notes
afmFileLookupRefreshInterval	0 through 2147483647	30	Yes	Yes	This is an optional attribute that is set by default when the fileset is created.
afmFileOpenRefreshInterval	0 through 2147483647	30	Yes	Yes	This is an optional attribute that is set by default when the fileset is created.
afmMode	**read-only** | **ro**, **local-updates** | **lu**, **single-writer** | **sw**, **independent-writer** | **iw**	**read-only** | **ro**	Not applicable	Not applicable	This is a required attribute for the **mmcrfileset** command.

Active file management settings

Once the NFS values are set, you can mount and access the AFM filesets. The first time the fileset is accessed the AFM NFS client mounts the home server or servers. To see these mounts on a gateway node, enter the following command:

```
cat /proc/mounts
```

The system displays the mount point and the mount options. If the **wsize** and **rsize** values are not 1MB, you can adjust the parameters and remount to get the correct values.

Tuning on the cache cluster

When the GPFS parameter **seqDiscardThreshold** has been set, it affects AFM as follows:
- If I/O requests are from a node other than the gateway node, there is no effect.
- If the read request is made on the gateway node for an uncached file, a higher **seqDiscardThreshold** value results in higher performance. This is because it allows the gateway to cache more data, which means that when the data is returned to the application, there is a greater chance that it comes out of the cache.

AFM performance tuning parameters at cache

The parameters in the following table can be used for performance tuning at cache:

Table 44. AFM configuration parameters that can be used for performance tuning at cache

AFM configuration parameter	Valid values	Default value	Tunable at cluster level	Tunable at fileset level
afmNumReadThreads	1 to 64	1	Yes	No
afmNumWriteThreads	1 to 64	1	Yes	No
afmParallelReadChunkSize	0 to 2147483647	128	Yes	Yes
afmParallelReadThreshold	0 to 2147483647	1024	Yes	Yes
afmParallelWriteChunkSize	0 to 2147483647	128	Yes	Yes
afmParallelWriteThreshold	0 to 2147483647	1024	Yes	Yes
afmReadSparseThreshold	0 to 2147483647	128	Yes	No

In addition, the following can be used:

afmHardMemThreshold

Sets a limit to the maximum amount of memory that AFM can use on each gateway node to record changes to the file system. After this limit is reached, the fileset goes into a 'dropped' state.

Exceeding the limit and the fileset going into a 'dropped' state due to accumulated pending requests might occur if -

- the cache cluster is disconnected for an extended period of time.
- the connection with the home cluster is on a low bandwidth.

Reboot the gateway node after you change the value.

afmAsyncDelay

Specifies (in seconds) the amount of time by which write operations are delayed (because write operations are asynchronous with respect to remote clusters). For write-intensive applications that keep writing to the same set of files, this delay is helpful because it replaces multiple writes to the home cluster with a single write containing the latest data. However, setting a very high value weakens the consistency of data on the remote cluster.

This configuration parameter is applicable only for writer caches (SW and IW), where data from cache is pushed to home.

Valid values are 1 through 2147483647. The default is 15.

System requirements and setup of a home and cache

Enabling a cache on an existing file system might involve some additional steps and possibly tuning.

Defining gateway nodes

Gateway nodes are needed on the cache sides to be able to communicate to home. An NFS client should be available on the gateway nodes, and they need to be able to mount the NFS-exported path of the home cluster.

When a gateway node is added to the cluster, it is initially inactive and will not be used to communicate with the home cluster. Once the GPFS daemon on the gateway node is restarted (or simply started if it was previously down), it is activated and is available for AFM functionality.

Use the **mmchnode** command to define a node as a gateway node. For example, to define `node1` as a gateway node, run the following command:

```
mmchnode --gateway –N node1
```

Adding or removing nodes

In essence, an application node is no different from a node in a standard GPFS file system. Therefore, the addition or removal of an application node is the same as the addition or removal of a standard GPFS node.

Setting up home and cache clusters

You can set up home and cache clusters and enable cache cleaning as follows.

Setting up the home cluster

NFS v3 is to be used at the home cluster.

When configuring a file system on the home cluster for external access, you must complete the following steps on all nodes that will be used to export the file system:

1. Start the file system.
2. Set NFS export information. Ensure that the following items are true:
   - The directory within the file system to be cached is exported and is configured with the correct access permissions.
   - Gateway nodes at cache site have access to the exported directory (can be mounted using NFS).

   Note: The **no_root_squash** and **sync** NFS export arguments are critical.

   For single-writer filesets, ensure that the file system is exported with **rw**. For read-only filesets, the file system can be exported with either **rw** or **ro**.
3. Start NFS services. This will start all necessary NFS programs and export the file system.

   Note: If the file system goes down for any reason, the exported file system must be re-exported.
4. Enable the exported path at home suitable for AFM, using the following command at the home cluster:

   mmafmconfig enable *ExportPath*

   See the **mmafmconfig** command description in the *IBM Spectrum Scale: Administration and Programming Reference*.

This is the GPFS startup sequence at home to prepare file system `gpfsA` (Linux):

```
# Start daemons
mmstartup -a

# Sample /etc/exports file contents:
/gpfs/gpfsA/ *(fsid=12345,rw,sync,insecure,no_root_squash,nohide,no_wdelay)

# Start nfs services
/etc/init.d/nfs start
```

Note: If both NFS and GPFS start automatically at bootup time, it is very important to ensure that GPFS starts before NFS. This is because NFS can only export GPFS if it is loaded. If NFS starts before GPFS, **exportfs -r** needs to be executed.

The home side of the relationship is defined at the NFS share level. The home contains all of the data available from the NFS mount point. It is a best practice to create one or more filesets for export on the home cluster, define NFS mount points at each fileset junction, and export each fileset individually.

Setting up a cache cluster

Before setting up a cache cluster, complete the following preparation steps:

1. Determine which nodes of the GPFS cluster will be application nodes and which will be gateway nodes.
2. Create an AFM fileset and link it.
3. Configure **maxfilestocache** to allow queueing of a large number of pending requests at gateway nodes.

Follow these steps to set up the cache cluster:

1. Prior to starting the GPFS daemon on the chosen gateway nodes, specify the gateway nodes:

   mmchnode --gateway -N *Node1*,*Node2*,**...**
2. Ensure that GPFS has been started:

   mmstartup -a

3. Mount the file system:
   ```
   mmmount Device
   ```
4. Create an AFM-enabled fileset. There is no special file system configuration required to be a cache. The cache relationship is defined when you create a fileset by specifying the **mmcrfileset -p** parameter and specifying the required attribute **afmTarget** and the optional **afmMode** attribute and **--inode-space** parameter.
   ```
   mmcrfileset Device Fileset -p afmTarget=Home-Exported-Path --inode-space=new
      -p afmMode=single-writer | read-only | local-updates | independent-writer
   ```
5. Link the fileset to the file system:
   ```
   mmlinkfileset device fileset -J /gpfs/gpfsA/fs1
   ```

Once the fileset is linked you are ready to start caching data.

Enabling cache cleaning

Cache cleaning is a feature where file data blocks are deallocated to make room for new files when fileset usage exceeds the fileset soft quota. This process is of deallocating blocks is called eviction. Note that the deallocation of file blocks is not done if file data is dirty and is not in sync with home. This feature is enabled only when fileset soft quotas are enabled.

To enable cache cleaning, enable a fileset soft quota for the cache fileset by specifying the **-Q** option on the **mmcrfs** or **mmchfs** commands. Cleaning starts when fileset usage reaches the soft quota limit.

Chapter 15. AFM-based Async Disaster Recovery

This topic describes the IBM Spectrum Scale AFM-based Async Disaster Recovery functionality.

Note: This feature is available only with IBM Spectrum Scale Advanced Edition.

Disaster Recovery solution is part of a larger overall business recovery/continuity solution. The goals include:

a) To provide business continuity in case of a disaster
b) To restore business continuance after a repair of a disaster
c) The ability to tolerate multiple disasters
d) The ability to minimize data loss in the event of a disaster

AFM-based Async Disaster Recovery is an AFM based fileset level replication disaster recovery capability to augment the overall business recovery solution. This capability is a strict one to one active-passive model. It is represented by two sites - primary and secondary.

The primary site is a read-writeable fileset where the applications are currently running and they have read-write access to the data. The secondary site is recommended to be read-only. All data from the primary site is asynchronously replicated to the secondary site.

Primary and secondary can be created independent of each other in terms of storage and network configuration. Once created, it is possible to establish a relationship between the two filesets. Primary is continuously available to the applications even in the event of communication failures or secondary failures. When the connection with the secondary is restored, primary automatically recognizes the restored connection and brings the secondary up to date, asynchronously in the background.

All file user data, metadata (including user extended attributes except inode number and atime), hard links, renames and clones from primary are replicated to the secondary. All file system and fileset related attributes such as user, group or fileset quotas, replication factors and dependent filesets from the primary are not replicated to the secondary.

A consistent point-in-time view of the data in the primary fileset can be propagated in-line to the secondary fileset with the use of fileset based snapshots (psnaps). The RPO (recovery point objective) feature makes it possible to specify the frequency of these snapshots and it can alert if it is unable to achieve the set RPO. Minimum RPO time is set to 15 minutes.

In the event of a disaster at the primary site, the secondary site can be promoted to act like a primary. If required, the secondary site's filesets can be restored to the state of last consistent RPO snapshot. Applications can be moved or failed over to this acting primary site. This helps to ensure business continuity in the event of a disaster with minimal downtime and minimal data loss.

AFM-based Async DR offers the capability to reconfigure the old primary or establish a new primary and synchronize it to the acting primary. This makes it possible for applications to eventually be failed back to the primary site as soon as the (new) primary is on the same level as the acting primary.

The following topics describe AFM-based Async DR capabilities and use case scenarios based on these capabilities with examples. Ensure that you adhere to all the steps for setup and functions described in the following topics.

All commands used in the steps in the following topics resume when rerun after interruptions or node failures.

AFM-based Async DR, restrictions are listed in the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html). The sections in AFM chapter such as Recovery, Disconnected operation, Parallel Writes, Peer Snapshots, Disabling AFM, and Tuning AFM Parameters will be relevant for primary and secondary filesets.

Introduction

This topic describes IBM Spectrum Scale AFM-based Async Disaster Recovery functionality.

Note: This feature is available only with IBM Spectrum Scale Advanced Edition.

Disaster Recovery solution is part of a larger overall business recovery/continuity solution. The goals include:

a) To provide business continuity in case of a disaster
b) To restore business continuance after a repair of a disaster
c) The ability to tolerate multiple disasters
d) The ability to minimize data loss in the event of a disaster

Async DR is an AFM based fileset level replication Disaster Recovery capability to augment the overall business recovery solution. This capability is a strict one to one active-passive model. It is represented by two sites - Primary and secondary.

The primary site is a read-writeable fileset where the applications are currently running and they have read-write access to the data. The secondary is recommended to be read-only and should not be used for direct writes under normal conditions. All data from the primary would be asynchronously replicated to the secondary.

Primary and secondary can be created independent of each other in terms of storage and network configuration. Once created, it is possible to establish a relationship between the two filesets. Primary is continuously available to the applications even in the event of communication failures or secondary failures. When the connection with the secondary is restored, primary will automatically recognize the restored connection and bring the secondary up to date, asynchronously in the background.

All file user data, metadata (including user extended attributes except inode number and atime), hard links, renames, clones from primary are replicated to the secondary. All file system/fileset related attributes such as user/group/fileset quotas, replication factors, dependent filesets from the primary are not replicated to the secondary.

A consistent point-in-time view of the data in the primary fileset can be propagated in-line to the secondary fileset with the use of fileset based snapshots (psnaps). The RPO feature makes it possible to specify the frequency of these snapshots and can alert if it is unable to achieve the set RPO. Minimum RPO time is set to 15 minutes.

In the event of a disaster at the primary site, the secondary can be promoted to act like a primary. If required, the secondary site's filesets can be restored to the state of last consistent RPO snapshot. Applications can be moved or failed over to this acting primary site. This helps to ensure business continuity in the event of disaster with minimal downtime and minimal data loss.

Async DR offers the capability to reconfigure the old primary or establish a new primary and sync it to the acting primary. This makes it possible for applications to eventually be failed back to the primary site as soon as the (new) primary is in sync with the acting primary.

Secondary is read-only by default. We can update secondary to be read-write using the **mmchconfig** command, **afmSecondaryRW** parameter.

This chapter will discuss each of these capabilities in theory and describe all use case scenarios with examples. All steps indicated in the following sections for setup and functions should be strictly adhered to.

All commands described in this chapter are reliable and will resume when re-run after interruptions or node failures.

While the AFM chapter would give a technical preview, restrictions are listed in the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html). The sections in AFM chapter such as Recovery, Disconnected operation, Parallel Writes, Peer Snapshots, Disabling AFM, Tuning AFM Parameters and HSM support will be relevant for primary and secondary filesets.

RTO

Recovery time objective (RTO) defines the amount of time it takes an application to fail over when a disaster occurs. Specific to AFM-based Async DR, RTO is the time taken for a fileset to become available for applications after a primary failure.

When a failover occurs, secondary is converted to acting primary. This can involve restoring the data from the last RPO snapshot which is the default, or if the **--norestore** option is used then the data in the secondary will be available as is. The second option is used if the application can tolerate the fact all the changes logged in the primary might not have been replicated to the secondary when the primary had a disaster. For more information, see “Failover to secondary” on page 246.

RTO can be estimated if the administrator has a rough idea of the amount of data generated in an RPO interval as this will be the amount of data which needs to restored from the last snapshot and it will also depend on the number of files which has changed in an RPO interval since a snapshot restore requires a complete scan of the snapshot to determine which files has changed.

In summary, fail over time is one of the components that the Administrator should use to calculate the RTO for the application.

Creating and using an AFM-based Async DR relationship

This topic describes how to create and use an AFM-based Async Disaster Recovery (DR) relationship.

You can create and use an AFM-based Async DR relationship as follows:

1. Create new primary fileset

   Create the primary fileset using the **mmcrfileset** command. The primary can be connected to the secondary using NFSv3 or the NSD protocol. All AFM parameters for writable filesets (single writer / independent writer) are applicable to a primary fileset. Like a Single Writer fileset a primary fileset does not check the secondary for changes because changes originate from the primary.

   A primary fileset is a writeable fileset so all file operations performed on this fileset are replayed at the secondary fileset using the same mechanism as single writer and independent writer modes. Unlike other AFM modes the secondary, or target fileset is an AFM fileset that has a relationship with a primary. Starting in Spectrum Scale 4.2 the secondary fileset is read-only. AFM parameters such as Async Delay, number of threads and parallel I/O can be used on primary filesets.

   When a primary fileset is created a unique primary id is generated. When creating the primary fileset you need to specify the path to the secondary fileset though it does not have to exist for the primary create to succeed. In this case the secondary has not yet been created but the path is provided in the **mmcrfileset** command.

```
# mmcrfileset fs1 primary2 -p afmMode=primary --inode-space=new -p afmTarget=nfs:///ibm/fs1/secondary2

Fileset primary2 created with id 19 root inode 7340035.
Primary Id (afmPrimaryId) 15997179941099568310-C0A8747F557F0086-19
```

2. 2. Create the secondary fileset

   The next step is to create the secondary fileset. To create the secondary you need the id of the primary fileset (primaryID). Use the **mmafmctl** command to get the **afmPrimaryId**.

   ```
   # mmafmctl fs1 getPrimaryId -j primary2
   Primary Id (afmPrimaryId) 15997179941099568310-C0A8747F557F0086-19
   ```

   Create the secondary fileset using the **mmcrfileset** command.

   ```
   mmcrfileset fs1 secondary2 -p afmMode=secondary –p
   afmPrimaryId=15997179941099568310-C0A8747F557F0086-19
    --inode-space new
   ```

3. Link the secondary fileset

   Link the secondary fileset using the **mmlinkfileset** command. If you are using NFS ensure, the NFS export on the secondary site is accessible from the Gateway nodes in the primary cluster. If you are using the NSD protocol, the secondary file system needs to be mounted on the Gateway nodes at the primary site.

   ```
   mmlinkfileset fs1 secondary2 -J /ibm/fs1/secondary2
   ```

   Data does not originate from the secondary so the primary does not check the secondary for changes. If you are using quotas make sure the quotas on primary and secondary are set to the same value. On a primary fileset eviction is disabled and filesets do not expire.

4. Link the primary fileset

   The next step is to link the primary fileset using the **mmlinkfileset** command. Linking the fileset creates the first RPO snapshot on the primary called `psnap0`.

   ```
   mmlinkfileset fs1 primary2 -J /ibm/fs1/primary2
   ```

   Once the primary and secondary are linked the RPO Snapshot (`psnap0`) is queued from the primary fileset then replayed on the secondary fileset. The Async DR filesets are now ready for use.

Converting GPFS filesets to primary or secondary

This topic describes how to convert GPFS filesets to primary or secondary sites.

You can convert GPFS filesets to primary or secondary as follows:

1. Ensure that primary and secondary sites have the same data using **trucking**.

   An existing GPFS independent fileset can be converted to primary or secondary. If the fileset on the primary site has data, the secondary site must be synchronized with the same data. This process is termed as trucking.

   If the fileset on the primary site has contents, synchronizing this with the secondary site can be done either using outband trucking or inband trucking.

   Outband trucking: Copying data manually using other ways such as **ftp**, **scp**, **rsync** etc. This must be completed before the relationship is established. Outband trucking can also be done by restoring the data on the secondary site from an existing backup of the primary site. It is the administrator's responsibility to ensure that the data restored on the secondary site is exactly the same as that on the primary. Extended attributes such as EA's and clone relationships should be exactly preserved on both sides. The conversion process assumes that the data on both sides are in synchronized and that it will not detect data or metadata conflicts.

   Inband trucking: Copying the data in the primary to the secondary while setting up the relationship. Inband trucking is limited by the network bandwidth between the primary and the secondary.

   Conversion of a regular independent fileset to AFM primary with **mmafmctl** command must be performed by specifying the **--check-metadata** option that verifies that the fileset does not contain objects with attributes that are disallowed in a primary fileset. These include:

   a. Files with **Immutable** and **AppendOnly** attributes
   b. Special files (such as devices)
   c. Dependent filesets
   d. File clones where the source belongs to a snapshot
2. Convert the fileset on the primary site to a primary using the mmafmctl command.

   It is recommended to have gateway nodes defined in the primary site and the file system mounted on all gateway nodes before doing this conversion.

   The command used for this purpose is the **mmafmctl** command **convertToPrimary** option:

   ```
   mmafmctl Device convertToPrimary -j FilesetName
      [ --afmtarget Target { --inband | --outband | --secondary-snapname SnapshotName }]
      [ --check-metadata | --nocheck-metadata ] [--rpo RPO] [-s LocalWorkDirectory]
   ```

   --afmtarget and **--inband** / **--outband** are mandatory options for the first conversion command on the fileset. If conversion got interrupted in the middle due to unforeseen reasons or in case of rare errors when the fileset is converted to a primary but left in PrimInitFail state, without creating the local psnap0, then the conversion command should be re-run without any argument. Alternatively, these can be converted back to normal GPFS filesets and converted again using the conversion command.

   The **--afmtarget** option mentions the fileset path on the secondary site.

   The **--inband** option is used for inband trucking. primary id gets generated and the first RPO snapshot psnap0 gets created. The entire data on the snapshot is queued to the secondary. Once the data has replayed on the secondary after Step 3 (following), that is, the primary and secondary are connected, it creates a psnap0 on the secondary ensuring that the psnap0 on the primary and the secondary are the same. At this point one can consider a relationship has been established.

   The **--outband** option is used for outband trucking. Primary id gets generated, a psnap0 gets created on the primary site and queued to the secondary. After the psnap0 is created on the secondary after Step 3 (following), that is, the primary and secondary are connected, it is important to ensure that the contents of psnap0 is same on both sides before this relationship is established.

   The option **--check-metadata** checks if the disallowed types (like immutable/append-only files, file clones where the source belongs to a snapshot etc) are present in the GPFS fileset on the primary site before the conversion. Conversion fails with this option if such files still exist. **--check-metadata** is not mandatory and performs a scan of the entire fileset to verify its contents, and while it can be excluded if the fileset is known to be permissible for conversion, it should be used when in doubt.

   The option **--nocheck-metadata** is used if one needs to proceed with conversion without checking for the disallowed types.

   The **--rpo** option specifies the RPO interval in minutes for this primary fileset.

   The **--secondary-snapname** is not applicable for converting AFM or GPFS filesets. This will be used while establishing a new primary, as discussed in subsequent sections.

   Gateway node assignments must be finalized and done preferably before conversion of GPFS or AFM filesets to primary. If no gateway is present on the primary cluster during conversion, then primary fileset might remain in the PrimInitFail state. For information on cache states, see “Cache states” on page 248.
3. Convert the fileset on the secondary site to a secondary and set the primary id using the **mmchfileset** or the **mmafmctl** command **convertToSecondary** option.

   Ensure that the NFS export on the secondary site is accessible on the primary, if NFS is used for defining AFM target on primary site. If GPFS protocol is used for the target, the secondary file system should be mounted on the primary site.

   Note: If you are establishing the secondary using the out-of-band option, you must first complete the data copy and ensure that the primary and the secondary have the same data before you configure the secondary with the primary's unique ID.

   Once the primary and secondary are connected with psnap0 on either side, the primary shows Active state. The two filesets are ready for use.

Converting AFM filesets to primary

This topic describes how to convert a working AFM single writer (SW) or independent writer (IW) relationship to a primary or secondary relationship.

A working AFM single writer (SW) or independent writer (IW) relationship can be converted to a primary/secondary relationship as follows:

1. Ensure all contents are cached

   It is recommended to have the fileset in active state by flushing queues and for filesets that have contents at home, the complete namespace should be constructed at the cache using stat on all entries, to avoid orphans.

   In an SW/IW it is possible that some files are not cached and/or some files have been evicted. All such files should be cached using prefetch and ensured that all contents are up to-date in the SW/IW cache. The following are some steps to ensure this:

   - Ensure that the storage capacity is same on the cache fileset as on the home and that any set quotas match.
   - Disable automatic eviction at cache using:

     `mmchfileset` *`filesystem sw/iw cache`* `-p afmEnableAutoEviction=no`
   - Ensure that **afmPrefetchThreshold** is set to 0 on the sw/iw cache.
   - Run a policy scan on the home to get the list of files and use that in the **mmafmctl prefetch** command on the cache to ensure all files are cached.
   - Alternately, run a policy scan on the cache to test the cached flag of each file and report on any that are not fully cached.
2. Convert the fileset on the primary site to a primary using the **mmafmctl** command.

   The primary id gets generated and a `psnap0` gets created on the primary site.

   It is recommended to have AFM gateway nodes defined in the primary site and that the file system is mounted on all gateway nodes before doing this conversion.

   The command used for this purpose is the **mmafmctl** command **convertToPrimary** option:

   ```
   mmafmctl Device convertToPrimary -j FilesetName
      [ --afmtarget Target { --inband | --outband | --secondary-snapname SnapshotName }]
      [ --check-metadata | --nocheck-metadata ] [--rpo RPO] [-s LocalWorkDirectory]
   ```

   The **--afmtarget** need not be used for AFM filesets as target would already have been defined.

   The **--inband** or **--outband** option are not to be used as both sides should be in sync before conversion.

   The **--check-metadata** option is used to detect immutable/appendonly files and orphans on the fileset before conversion. The conversion command works using a policy to check this and conversion fails if such files still exist. SW/IW filesets should have established contact with home at least once for this option to succeed.

   The **--nocheck-metadata** option is used if one needs to proceed with conversion without checking for immutable/appendonly files and orphans.

   If there is activity on the filesets during conversion, the policy is likely to detect orphans in the fileset. In that case, the command has to be run again. If the problem does not go away even after repetitive execution of the conversion command on a live fileset, the **--check-metadata** option can be used to do the conversion.

   The **--secondary-snapname** is not applicable for converting AFM/GPFS filesets. This will be used while establishing a new primary, as discussed in subsequent sections.

   The **--rpo** option specifies the RPO interval in minutes for this primary fileset.

   In case of multiple IW filesets pointing to the same home, only one IW fileset can be converted to primary. Before starting the conversion process, all remaining IW filesets should be made to point to a new home or should be converted as non-AFM filesets using **mmchfileset** command with **-p**

afmTarget=disable option. The one IW fileset that is to be converted should have all data from the home cached and directories complete before conversion.

Only SW/IW filesets in active/inactive, dirty and unmounted/disconnected states can be converted. Conversion is not allowed in any other fileset state such as `FailoverInProgress`, `FailbackInProgress`, `Recovery`, `Dropped`, `NeedsResync`, `QueueOnly`, `FlushOnly` and `FailbackComplete`.

3. Convert the home to a secondary and set the primary id using the **mmchfileset** command or the **mmafmctl** command **convertToSecondary** option.

   Once the primary and secondary are converted and connected through primary Id, in the preceding steps, the `psnap0` queued from the primary fileset is played on the secondary fileset. The two filesets are ready for use.

RPO snapshots

Recovery point objective (RPO) snapshots are used to maintain consistency between primary and secondary filesets.

The default RPO interval is 15 minutes and can be updated using the `afmRPO` parameter during primary creation time or during the course of operation between primary and secondary filesets. This interval indicates the amount of data loss which can tolerated in the event of failure.

The appropriate RPO intervals for each application setup are guided by factors such as rate of data change, fileset parameters such as **`afmAsyncDelay`** and **`afmNumFlushThreads`**, network bandwidth between the two sites, the maximum tolerable data loss in the event of disaster, etc. Each RPO interval triggers a fileset or cache level snapshot which results in the file system being quiesced. Quiescing the file system is an I/O intensive action, therefore at a performance optimization, if multiple caches have similar RPOs, then their snapshots are batched together so that the file system is quiesced once. As a further optimization, RPO snapshots are only taken for primary filesets that have experienced data or metadata change since the last RPO period.

The **afmAsyncDelay** parameter specifies the minimum time the cache must wait before flushing pending I/O to the secondary. An RPO request in queue flushes the queue before creating a snapshot on the secondary site. The general guideline is to set async delay less than the RPO interval.

An RPO miss event - The AFM_RPO_MISS event can occur if RPO is missed due to network delay or failure to create on the secondary site. Failed RPOs are re-queued on the gateway and are re-tried at the secondary site.

At any point in time, two RPO snapshots apart from `psnap0` are retained on both sides. Deletions from primary are done and queued when new RPO snapshots arrive. While there is every attempt to ensure that deletes are successful in secondary, there can be some extra snapshots due to some deletes not succeeding. In such cases, you need to manually delete those extra snapshots on secondary using the **mmdelsnapshot -p** option.

Apart from automatic RPO snapshots, you can create user snapshots using the **mmpsnap create** command with the **-rpo** option. These RPOs do not get deleted as part of a subsequent RPO snapshot delete process.

When a failure occurs on the primary you might need to roll back the last consistent snapshot present on the secondary while converting it into acting primary. When applications are moved to the acting primary, RPO snapshots are not taken. Therefore, RPO is not applicable on acting primary.

The gateway node mapped to serve a fileset is called the metadata server (MDS) of the fileset. The MDS acts as the owner for the fileset. If the MDS goes experiences failure, another gateway node takes over the fileset as MDS. Recovery gets triggered on the new MDS while taking over the fileset. When recovery gets triggered when the next RPO snapshot is due on the fileset, this RPO snapshot is used for recovery.

While recovery RPO snapshots are played to home as usual, the regular RPO snapshot deletes are not scheduled in these times. Hence, some extra RPO snapshots temporarily appear on the primary and secondary, which are cleaned up on subsequent RPO intervals.

All user created and system created RPO snapshots except the `psnap0` belonging to an active primary-secondary will have to be deleted by user before a primary-secondary is deleted.

You can change the RPO interval at any time with the **mmchfileset** command. The time for the next RPO snapshot to occur (Tnext) is calculated by adding the RPO interval that you set (Irpo) to the time at which the previous snapshot occurred (Tprev): `Tnext = Tprev + Irpo`. If no file in the RPO snapshot changes during the interval, then GPFS does not take the snapshot, and the time for the next RPO snapshot is calculated as `Tnext = Tnext + Irpo`.

Notice that the RPO interval is not added to the time at which you set the RPO interval, but rather to the time of the last RPO snapshot. For example, assume that the previous snapshot was at 9:00 AM, that the RPO interval is 30 minutes, and that the current time is 9:15 AM. If you now change the RPO interval to 60 minutes, then the next RPO snapshot occurs at 10:00 AM (that is, 9:00 + 60 minutes) rather than at 10:15 AM (9:15 AM + 60 minutes).

The RPO management deletes the oldest snapshot when it creates a new one but it never deletes the `psnap0`. **`mmpsnap`** command would allow you to delete `psnap0`. By deleting `psnap0` storage utilization can be improved as we do not hold down the data blocks used by `psnap0` which can be significant over a period of time. Also, deletion of `psnap0` can improve the performance of subsequent creation and deletion of RPO snapshots. However, `psnap0` deletion should be done very carefully using discretion and only when there are other RPO snapshots present in the primary/secondary filesets to handle disaster recovery. If `psnap0` is deleted without any other RPO snapshots you can end loosing data which will have to be recreated.

Failover to secondary

When the primary fileset experiences a disaster, all applications need to be moved to the secondary fileset to ensure business continuity.

When primary experiences a disaster, all applications need to be moved to the secondary to ensure business continuity. The secondary must be first converted to an acting primary using the **mmafmctl** command **failoverToSecondary** option.

```
mmafmctl Device failoverToSecondary -j FilesetName [--norestore |--restore ]
```

There is a choice of restoring the latest snapshot data on the secondary during the failover process or leave the data as is using the **--norestore** option. Once this is complete, the secondary becomes ready to host applications. The customer also needs to ensure that the NFS export used for the secondary is not removed so that on failback to the primary, the old primary is able to communicate with the secondary or acting primary.

RPO snapshots are temporarily disabled on the acting primary. The new data is not replicated to another site until a new primary is set up or the old primary is reinstalled and this acting primary is converted back to secondary. This is described in the subsequent topics.

Failing back to old primary

This topic describes the process for failing back to the old primary after disaster recovery.

You can reinstall the old primary when it is repaired and it is back online after the disaster as follows:

1. While the applications continue to run on the acting primary, start configuring the old primary using the **mmafmctl** command **failbackToPrimary--start** option.

```
mmafmctl Device failbackToPrimary -j FilesetName { --start | --stop [--force] }
```

The **--start** option restores the primary to the contents from the last RPO on the primary before the disaster. It is assumed that since old primary came back up, all RPOs prior to the disaster are present and accessible.

If this is not the case and the first RPO snapshot `psnap0` is lost, the old Primary can be converted to a normal GPFS fileset, and the steps described in "Failing back to new primary" must be followed to set it up.

The **--start** option puts the primary in the read-only mode, to avoid accidental corruption until the fail back process is completed.

2. Apply differences created by applications on the old primary, since acting primary took over the applications using the **mmafmctl** command **applyUpdates** option.

   ```
   mmafmctl Device { applyUpdates | getPrimaryId } -j FilesetName
   ```

   All the differences can be brought over in one go or through multiple iterations. For minimizing application downtime, this command can be run multiple times to bring the old primary's contents in sync with the acting primary. When the contents on both the sites are as close as possible or are minimal, applications must take a downtime and then this command must be run one last time.

   It is possible that **applyUpdates** fails with an error during instances when the acting primary is overloaded. In such cases the command needs to be run again.

3. Complete the failback process using the **mmafmctl** command **failbackToPrimary --stop** option.

   This command puts the fileset in the read-write mode. The primary is now ready for starting applications.

   If failback's **--stop** option does not complete due to errors and it does not allow for failback to be stopped, it can be forced to stop with the **--force** option.

4. Convert the acting primary back to secondary and set the primary id.

   Unlink the acting primary, change it back to secondary, followed by linking the secondary back. NFS can be restarted on the secondary side, to ensure that the secondary export is accessible to the primary.

   The primary and secondary are connected back as before the primary disaster and all data from the primary will be played on the secondary as earlier. Regular RPO also resume on the primary.

Failing back to new primary

This topic describes the process for failing back to new primary fileset after disaster recovery.

You can configure a new primary if the old primary is completely lost after the disaster as follows.

1. Create a new GPFS fileset on the new primary site.

   This is configured as the new primary.

2. Create the latest snapshot from the acting primary using the **mmafmctl** command **replacePrimary** option.

   This creates a new `psnap0` with all the latest contents on acting primary. This snapshot is used in the subsequent steps in this task.

3. Copy the contents from the snapshot created in the preceding step to the new primary using **scp** or other means outside of AFM.

4. Convert the fileset on the primary site to a primary, using the **mmafmctl** command **convertToPrimary --outband** and **--secondary-snapname** options.

   The snapshot created in step 2 is given as argument to **--secondary-snapname** option.

   ```
   mmafmctl Device convertToPrimary -j FilesetName
        [ --afmtarget Target { --inband | --outband | --secondary-snapname SnapshotName }]
        [ --check-metadata | --nocheck-metadata ] [--rpo RPO] [-s LocalWorkDirectory]
   ```

   Primary id gets generated and a `psnap0` with the same name gets created on the primary site.

After this, run all steps in "Failing back to old primary" on page 246

Changing secondary

This topic describes how to create a new secondary after a disaster.

A disaster at the secondary can take place resulting in secondary no longer being available. The recovery steps are as follows:

1. Create a new GPFS independent fileset on the new secondary site.

   On the new secondary site a new GPFS independent fileset has to be created.
2. Trucking data

   Data on the primary can be copied to the new GPFS fileset, created in the preceding step, using other means such as ftp, scp etc. Alternatively it can be decided that data will be trucked using the relationship. In this case, there is nothing to do in this step.
3. Establish a new secondary for the primary using the **mmafmctl** command **changeSecondary** option.

   ```
   mmafmctl Device changeSecondary -j FilesetName
   --new-target NewAfmTarget [ --target-only |--inband | --outband ]
           [-s LocalWorkDirectory]
   ```

   --inband or **--outband** option is used based on the method used to truck data in step 2. The **--target-only** option is used if we need to change the NFS server or IP address of the secondary. This can also be used to change to a new NFS server on the secondary site. The old and new NFS servers should be of the same architecture.
4. Convert the GPFS fileset on the new secondary site to a secondary and set the primary id using the **mmchfileset** or the **mmafmctl** command **convertToSecondary** option.

   Ensure that the NFS export on the secondary site is accessible on the primary, if NFS is used for defining AFM target on primary site. If GPFS protocol is used for the target, the secondary file system must be mounted on the primary site.

   Once the primary and secondary are linked, in the preceding steps, the `psnap0` queued from the primary fileset is played on the secondary fileset. The new secondary is now linked to this primary.

Cache states

This topic describes the allowable cache states of the primary fileset.

The cache states of primary fileset can be any of the following:

PrimInitInProg
: This state is used while primary and secondary are in the process of establishing a relationship while the `psnap0` is in progress. All operations are disallowed till `psnap0` is taken locally.

PrimInitFail
: This is a rare failure state when the `psnap0` has not been taken at the primary. In this state no data is moved from the primary to the secondary. The administrator should check that the gateway nodes are up and file system is mounted on them on the primary. The secondary fileset should also be setup correctly and available for use. Re-running the **convertToPrimary** command without any parameters helps to come out of this state.

Active The primary is ready for use.

Dirty The primary has requests in queue.

NeedsResync
: This is a rare state and is possible only under error conditions during recovery, failback or

conversion. If this error is during a conversion or recovery, the problem should get automatically fixed in the subsequent recovery. If the error is during failback, then the failback process should be run again.

Dropped
This is a rare state and is possible only under certain error conditions. The possible conditions are as listed in Chapter 14, “Active file management,” on page 213 for this state (except IW failback).

FailbackInProg
When failback start has been initiated and failback is in progress.

Chapter 14, “Active file management,” on page 213 lists explanations for the following states for primary filesets:

`Disconnected`, `Unmounted`, `Recovery`, `FlushOnly`, `QueueOnly`, `Dropped`, `NeedsResync`, `Inactive`

HSM support on AFM DR filesets

This topic describes HSM support on AFM DR filesets.

HSM is supported on AFM DR filesets. Dirty files are not migrated from primary. Include the following parameter in `dsm.opt` file to skip dirty files:

```
AFMSKIPUNCACHEDFILES yes
```

Also, if the DR Secondary fileset is configured to use HSM, then the environment variable *DSM_LOG* should be defined to point to different directory out side of DR Secondary fileset, where `dsmerror.log` file can be created else TSM would fail to create `dsmerror.log` file in DR Secondary since DR Secondary is a read-only fileset.

AFM-based Async DR use case 1: Creating new primary-secondary filesets, check RPOs creation on expiry of RPO timeout, failing over and failing back to old primary

This use case scenario describes creating new primary-secondary filesets, checking RPO's creation on expiry of RPO timeout, failing over and failing back to the old primary fileset.

For this usecase scenario, the secondary's file system is to be remote mounted via cross-cluster mount.

1. Create primary using the **mmcrfileset** command::

```
mmcrfileset fs1 drp12 --inode-space=new -p afmtarget=gpfs:///gpfs/remotefs1/drs12 -p afmmode=drp --inode-limit=1024000 -p afmAsyncDelay=15 -p afmRPO=15
Fileset drp12 created with id 23 root inode 7864323.
Primary Id (afmPrimaryId) 14228043454022319638-C0A8037555D2994D-23


Primary:
mmlsfileset fs1 drp12 -L --afm
Filesets in file system 'fs1':

Attributes for fileset drp12:
==============================
Status                                  Unlinked
Path                                    --
Id                                      23
Root inode                              7864323
Parent Id                               --
Created                                 Tue Aug 25 01:16:06 2015
Comment
Inode space                             15
Maximum number of inodes                1024000
Allocated inodes                        500736
Permission change flag                  chmodAndSetacl
afm-associated                          Yes
Target                                  gpfs:///gpfs/remotefs1/drs12
Mode                                    primary
Async Delay                             15
Recovery Point Objective                15 minutes
Last pSnapId                            0
Number of Gateway Flush Threads         4
Primary Id                              14228043454022319638-C0A8037555D2994D-23
```

2. Create secondary on the secondary cluster:

```
/usr/lpp/mmfs/bin/mmcrfileset fs1 drs12 --inode-space=new --inode-limit=1024000 -p afmMode=drs -p afmPrimaryId=14228043454022319638-CA8037555D2994D-23
Fileset drs12 created with id 43 root inode 11010051.

/usr/lpp/mmfs/bin/mmlinkfileset fs1 drs12 -J /fs1/drs12
Fileset drs11 linked at /fs1/drs12

/usr/lpp/mmfs/bin/mmlsfileset fs1 drs12 -L --afm
Filesets in file system 'fs1':

Attributes for fileset drs12:
==============================
Status                                  Linked
Path                                    /fs1/drs12
Id                                      43
Root inode                              11010051
Parent Id                               0
Created                                 Tue Aug 25 01:26:51 2015
Comment
Inode space                             21
Maximum number of inodes                1024000
Allocated inodes                        501504
Permission change flag                  chmodAndSetacl
afm-associated                          Yes
Associated Primary ID                   14228043454022319638-C0A8037555D2994D-23
Mode                                    secondary
Last pSnapId                            0
```

3. Link primary to create `psnap0`:

   ```
   mmlinkfileset fs1 drp12 -J /fs1/drp12
   Fileset drp12 linked at /fs1/drp12
   First snapshot name is psnap0-rpo-C0A8037555D2994D-23
   Flushing dirty data for snapshot drp12::psnap0-rpo-C0A8037555D2994D-23...
   Quiescing all file system operations.
   Snapshot drp12::psnap0-rpo-C0A8037555D2994D-23 created with id 36.
   ```

   Primary State:

```
mmafmctl fs1 getstate -j drp12
Fileset Name  Fileset Target              Cache State    Gateway Node   Queue Length   Queue numExec
------------  --------------             -------------   ------------   ------------   -------------
drp12        gpfs:///gpfs/remotefs1/drs12   Active         c3m3n06          0             2
```

4. Create data from primary and see that it goes to secondary:

   ```
   /usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drp12/file_pri_1
     recSize 10K nBytes 100M fileSize 100M
     nProcesses 1 nThreadsPerProcess 1
     file cache flushed before test
     not using direct I/O
     offsets accessed will cycle through the same file segment
     not using shared memory buffer
     not releasing byte-range token after open
     no fsync at end of test
       Data rate was 24599.12 Kbytes/sec, thread utilization 0.979

   /usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drp12/file_pri_2
     recSize 10K nBytes 100M fileSize 100M
     nProcesses 1 nThreadsPerProcess 1
     file cache flushed before test
     not using direct I/O
     offsets accessed will cycle through the same file segment
     not using shared memory buffer
     not releasing byte-range token after open
     no fsync at end of test
       Data rate was 25954.27 Kbytes/sec, thread utilization 0.999
   ```

   Primary contents:

   ```
   ls -l /fs1/drp12
   total 204800
   -rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
   -rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
   ```

   Secondary contents:

```
ls -l /fs1/drs12
total 409600
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

5. Check RPO after few mins:

```
mmlssnapshot fs1 -j drp12
Snapshots in file system fs1:
Directory                SnapId    Status  Created                   Fileset
psnap0-rpo-C0A8037555D2994D-23 36        Valid   Tue Aug 25 01:27:45 2015  drp12
psnap-rpo-C0A8037555D2994D-23-15-08-25-01-41-51 37        Valid   Tue Aug 25 01:41:56 2015  drp12
```

Create more data to primary:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drp12/file_pri_3
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 21322.77 Kbytes/sec, thread utilization 0.978

/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drp12/file_pri_4
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 21617.21 Kbytes/sec, thread utilization 1.000
```

Primary State:

```
mmafmctl fs1 getstate -j drp12
Fileset Name   Fileset Target               Cache State     Gateway Node   Queue Length   Queue numExec
------------   --------------             -------------    ------------   ------------   -------------
drp12        gpfs:///gpfs/remotefs1/drs12       Dirty          c3m3n06            4          40963
```

Note: At this point the 2nd RPO snapshot has not yet triggered.

6. Unlink primary feigning primary going down:

```
/usr/lpp/mmfs/bin/mmunlinkfileset fs1 drp12 -f
Fileset drp12 unlinked.
```

7. Failover - convert secondary to acting primary. This is to be run on the secondary.

```
/usr/lpp/mmfs/bin/mmafmctl fs1 failoverToSecondary -j drs12
mmafmctl: failoverToSecondary restoring from psnap psnap-rpo-C0A8037555D2994D-23-15-08-25-01-41-51
[2015-08-25 01:48:07] Restoring fileset "drs12" from snapshot "psnap-rpo-C0A8037555D2994D-23-15-08-25-01-41-51" of filesystem "/dev/f1"

[2015-08-25 01:48:09] Scanning inodes, phase 1 ...
[2015-08-25 01:48:25] 11511552 inodes have been scanned, 50% of total.
[2015-08-25 01:48:41] 23023104 inodes have been scanned, 100% of total.
[2015-08-25 01:48:41] Constructing operation list, phase 2 ...
[2015-08-25 01:48:41] 0 operations have been added to list.
[2015-08-25 01:48:41] 2 operations have been added to list.
[2015-08-25 01:48:41] Deleting the newly created files, phase 3 ...
[2015-08-25 01:48:42] Deleting the newly created hard links, phase 4 ...
[2015-08-25 01:48:43] Splitting clone files, phase 5 ...
[2015-08-25 01:48:43] Deleting the newly created clone files, phase 6 ...
[2015-08-25 01:48:44] Moving files, phase 7 ...
[2015-08-25 01:48:45] Reconstructing directory tree, phase 8 ...
[2015-08-25 01:48:46] Moving files back to their correct positions, phase 9 ...
[2015-08-25 01:48:46] Re-creating the deleted files, phase 10 ...
[2015-08-25 01:48:47] Re-creating the deleted clone parent files, phase 11 ...
[2015-08-25 01:48:48] Re-creating the deleted clone child files, phase 12 ...
[2015-08-25 01:48:49] Re-creating the deleted hard links, phase 13 ...
[2015-08-25 01:48:50] Restoring the deltas of changed files, phase 14 ...
[2015-08-25 01:48:50] Restoring the attributes of files, phase 15 ...
[2015-08-25 01:48:51] Restore completed successfully.
```

```
[2015-08-25 01:48:51] Clean up.
Primary Id (afmPrimaryId) 5802564250705647455-C0A8286F55B0E3EE-43
Fileset drs12 changed.
Promoted fileset drs12 to Primary
```

Acting primary:

```
/usr/lpp/mmfs/bin/mmlsfileset fs1 drs12 -L --afm
Filesets in file system 'fs1':

Attributes for fileset drs12:
==============================
Status                                  Linked
Path                                    /fs1/drs12
Id                                      43
Root inode                              11010051
Parent Id                               0
Created                                 Tue Aug 25 01:26:51 2015
Comment
Inode space                             21
Maximum number of inodes                1024000
Allocated inodes                        501504
Permission change flag                  chmodAndSetacl
afm-associated                          Yes
Target                                  --
Mode                                    primary
Async Delay                             15 (default)
Recovery Point Objective                disable
Last pSnapId                            0
Number of Gateway Flush Threads         4
Primary Id                              5802564250705647455-C0A8286F55B0E3EE-43
```

Acting primary contents after failover:

```
ls -l /fs1/drs12
total 409600
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

8. Create sample data from acting primary:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drs12/file_actingpri_1
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 48282.25 Kbytes/sec, thread utilization 0.991

/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drs12/file_actingpri_2
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 58104.41 Kbytes/sec, thread utilization 0.999
```

Contents from acting primary:

```
ls -l /fs1/drs12/
total 812544
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
```

```
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

9. Link old primary when it is back:

```
mmlinkfileset fs1 drp12 -J /fs1/drp12
Fileset drp12 linked at /fs1/drp12
```

10. Run failback start on old primary:

```
mmafmctl fs1 failbackToPrimary -j drp12 --start
Fileset drp12 changed.
mmafmctl: failbackToPrimary restoring from psnap psnap-rpo-C0A8037555D2994D-23-15-08-25-01-41-51
[2015-08-25 01:50:52] Restoring fileset "drp12" from snapshot "psnap-rpo-C0A8037555D2994D-23-15-08-25-01-41-51" of filesystem "/dev/f1"

[2015-08-25 01:50:54] Scanning inodes, phase 1 ...
[2015-08-25 01:51:03] 8365056 inodes have been scanned, 50% of total.
mmlssn[2015-08-25 01:51:12] 16730112 inodes have been scanned, 100% of total.
[2015-08-25 01:51:12] Constructing operation list, phase 2 ...
[2015-08-25 01:51:12] 0 operations have been added to list.             2015-08-25 01:51:12] 2 operations have been added to list.
[2015-08-25 01:51:12] Deleting the newly created files, phase 3 ...        [201-08-25 01:51:13] Deleting the newly created hard links, phase 4 ...
[2015-08-25 01:51:13] Splitting clone files, phase 5 ...
[2015-08-25 01:51:14] Deleting the newly created clone files, phase 6 ...
[2015-08-25 01:51:15] Moving files, phase 7 ...
[2015-08-25 01:51:16] Reconstructing directory tree, phase 8 ...
[2015-08-25 01:51:16] Moving files back to their correct positions, phase 9 ...
[2015-08-25 01:51:17] Re-creating the deleted files, phase 10 ...
[2015-08-25 01:51:18] Re-creating the deleted clone parent files, phase 11 ...
[2015-08-25 01:51:18] Re-creating the deleted clone child files, phase 12 ...
[2015-08-25 01:51:19] Re-creating the deleted hard links, phase 13 ...
[2015-08-25 01:51:20] Restoring the deltas of changed files, phase 14 ...
[2015-08-25 01:51:21] Restoring the attributes of files, phase 15 ...
[2015-08-25 01:51:21] Restore completed successfully.
[2015-08-25 01:51:21] Clean up.
```

Primary contents:

```
ls -l /fs1/drp12
total 204800
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

11. Create more data on acting primary:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drs12/file_actingpri_3
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 51110.26 Kbytes/sec, thread utilization 0.996
```

12. Run **applyUpdates** to sync up primary to acting primary:

```
mmafmctl fs1 applyUpdates -j drp12

[2015-08-25 01:51:39] Getting the list of updates from the acting Primary...
[2015-08-25 01:52:21] Applying the 9 updates...
[2015-08-25 01:52:25] 9 updates have been applied, 100% of total.
mmafmctl: Creating the failback psnap locally. failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38
Flushing dirty data for snapshot drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38...
Quiescing all file system operations.
Snapshot drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38 created with id 38.
```

Primary contents:

```
ls -l /fs1/drp12
total 512000
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:51 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

13. Create more data on acting primary to show applications continue and then applications stop:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drs12/file_actingpri_4
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 54504.44 Kbytes/sec, thread utilization 0.991
```

Acting primary contents:

```
ls -l /fs1/drs12
total 1227264
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:51 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Aug 25 01:52 file_actingpri_4
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

14. Run last **applyUpdates** on old primary after applications stop.

```
mmafmctl fs1 applyUpdates -j drp12
[2015-08-25 01:52:43] Getting the list of updates from the acting Primary...
[2015-08-25 01:53:25] Applying the 3 updates...
[2015-08-25 01:53:26] 3 updates have been applied, 100% of total.
mmafmctl: Creating the failback psnap locally. failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-52-42
Flushing dirty data for snapshot drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-52-42...
Quiescing all file system operations.
Snapshot drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-52-42 created with id 39.
mmafmctl: Deleting the old failback psnap. failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38
Invalidating snapshot files in drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38...
Deleting files in snapshot drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38...
 100.00 % complete on Tue Aug 25 01:53:26 2015  (    500736 inodes with total         0 MB data processed)
Invalidating snapshot files in drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38/F/...
Delete snapshot drp12::failback-psnap-rpo-C0A8037555D2994D-23-15-08-25-01-51-38 complete, err = 0
```

Primary contents:

```
ls -l /fs1/drp12
total 614400
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:51 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Aug 25 01:52 file_actingpri_4
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

15. Complete failback process on old primary:

```
mmafmctl fs1 failbackToPrimary -j drp12 --stop
Fileset drp12 changed.
```

16. Convert the acting primary back to secondary and re-establish the relationship:

```
/usr/lpp/mmfs/bin/mmunlinkfileset fs1 drs12 -f
Fileset drs12 unlinked.

/usr/lpp/mmfs/bin/mmchfileset fs1 drs12 -p afmmode=drs,
afmPrimaryId=14228043454022319638-C0A8037555D2994D-23
Fileset drs12 changed.


/usr/lpp/mmfs/bin/mmlinkfileset fs1 drs12 -J /fs1/drs12
Fileset drs11 linked at /fs1/drs12
```

Primary contents:

```
ls -l /fs1/drp12
total 614400
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:51 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Aug 25 01:52 file_actingpri_4
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

Secondary contents:

```
ls -l /fs1/drs12
total 1228800
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:51 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Aug 25 01:52 file_actingpri_4
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
```

17. Create data from failed back primary:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drp12/file_pri_5
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 21235.12 Kbytes/sec, thread utilization 0.985

/usr/lpp/mmfs/samples/perf/gpfsperf create seq /fs1/drp12/file_pri_6
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 22658.18 Kbytes/sec, thread utilization 1.000
```

Primary contents:

```
ls -l /fs1/drp12
total 819200
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:51 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Aug 25 01:52 file_actingpri_4
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:55 file_pri_5
-rw-r--r-- 1 root root 104857600 Aug 25 01:55 file_pri_6
```

Secondary contents:

```
ls -l /fs1/drs12
total 1638400
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:49 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:51 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Aug 25 01:52 file_actingpri_4
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_1
-rw-r--r-- 1 root root 104857600 Aug 25 01:29 file_pri_2
-rw-r--r-- 1 root root 104857600 Aug 25 01:55 file_pri_5
-rw-r--r-- 1 root root 104857600 Aug 25 01:55 file_pri_6
```

AFM-based Async DR use case 2: Converting GPFS filesets to primary-secondary using inband option, failing over and failing back to a new primary using outband trucking.

This use case scenario describes the conversion of GPFS filesets to primary-secondary using the inband option, and failover and failback to a new primary using outband trucking.

For this usecase scenario, the secondary's file system is to be remote mounted via cross-cluster mount.

1. Create GPFS fileset at primary site with contents:

```
mmcrfileset fs1 gpfs1022 --inode-space=new
Fileset gpfs1022 created with id 185 root inode 49283075.
mmlinkfileset fs1 gpfs1022 -J /gpfs/fs1/gpfs1022
Fileset gpfs1022 linked at /gpfs/fs1/gpfs1022

/usr/lpp/mmfs/samples/perf/gpfsperf create seq /gpfs/fs1/gpfs1022/file_gpfspri_1
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using data shipping
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 73161.03 Kbytes/sec, thread utilization 0.996

/usr/lpp/mmfs/samples/perf/gpfsperf create seq /gpfs/fs1/gpfs1022/file_gpfspri_2
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using data shipping
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 73639.84 Kbytes/sec, thread utilization 1.000
```

   Contents on the GPFS fileset:

   ```
   ls -l /gpfs/fs1/gpfs1022
   total 402944
   -rw-r--r-- 1 root root 104857600 Apr 28 03:52 file_gpfspri_1
   -rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
   ```

2. Create empty GPFS fileset at secondary site:

```
/usr/lpp/mmfs/bin/mmcrfileset homefs1 gpfs1022 --inode-space=new
Fileset gpfs1022 created with id 190 root inode 1345847299.

/usr/lpp/mmfs/bin/mmlinkfileset homefs1 gpfs1022 -J /gpfs/homefs1/gpfs1022
Fileset gpfs1022 linked at /gpfs/homefs1/gpfs1022

ls -l /gpfs/homefs1/gpfs1022
total 0
```

3. Convert the GPFS fileset at primary site to primary using **--inband** option:

```
/usr/lpp/mmfs/bin/mmafmctl fs1 convertToPrimary -j gpfs1022 --afmtarget=gpfs:///gpfs/remotefs1/gpfs1022 -inband
Checking for any special files. This may take a while...
Converting GPFS fileset to AFM primary fileset...
Primary Id (afmPrimaryId) 3235816757204815584-C0A802135530BD89-185
Fileset gpfs1022 changed.
Setting up a Primary and Secondary relation...
Data will be moved to secondary via AFM
psnap will be taken at secondary after data movement completes
```

4. Convert the GPFS fileset at secondary site to secondary and establish a relationship:

```
/usr/lpp/mmfs/bin/mmchfileset homefs1 gpfs1022 -p afmmode=secondary -p afmPrimaryId=3235816757204815584-C0A802135530BD89-185
Fileset gpfs1022 changed.
```

Primary contents:

```
ls -l /gpfs/fs1/gpfs1022
total 409600
-rw-r--r-- 1 root root 104857600 Apr 28 03:52 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
```

Secondary contents:

```
ls -l /gpfs/homefs1/gpfs1022
total 409600
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
```

5. Create data from primary and see that it goes to secondary:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /gpfs/fs1/gpfs1022/file_gpfspri_3
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using data shipping
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 35139.11 Kbytes/sec, thread utilization 0.996

/usr/lpp/mmfs/samples/perf/gpfsperf create seq /gpfs/fs1/gpfs1022/file_gpfspri_4
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using data shipping
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 35568.11 Kbytes/sec, thread utilization 0.999
```

Primary contents:

```
ls -l /gpfs/fs1/gpfs1022
total 819200
-rw-r--r-- 1 root root 104857600 Apr 28 03:52 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_4
```

Secondary contents:

```
ls -l /gpfs/homefs1/gpfs1022
total 614400
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_4
```

Create user RPO:

```
mmpsnap fs1 create -j gpfs1022 --rpo
Flushing dirty data for snapshot gpfs1022::psnap-rpo-C0A802135530BD89-185-15-04-28-03-54-37...
Quiescing all file system operations.
Snapshot gpfs1022::psnap-rpo-C0A802135530BD89-185-15-04-28-03-54-37 created with id 331.
```

6. Unlink primary feigning primary going down:

```
/usr/lpp/mmfs/bin/mmunlinkfileset fs1 gpfs1022 -f
Fileset gpfs1022 unlinked.
```

7. Failover - convert secondary to acting primary:

```
/usr/lpp/mmfs/bin/mmafmctl homefs1 failoverToSecondary -j gpfs1022
mmafmctl: failoverToSecondary restoring from psnap psnap-rpo-C0A802135530BD89-185-15-04-28-03-54-37
[2015-04-28 03:54:40] Restoring fileset "gpfs1022" from snapshot "psnap-rpo-C0A802135530BD89-185-15-04-28-03-54-37" of
filesystem "/dev/homefs1"
[2015-04-28 03:54:42] Scanning inodes, phase 1 ...
[2015-04-28 03:54:43] 1345947328 inodes have been scanned, 100% of total.
[2015-04-28 03:54:43] There's no data changes since the restoring snapshot, skipping restore.
[2015-04-28 03:54:43] Restore completed successfully.
[2015-04-28 03:54:43] Clean up.
Primary Id (afmPrimaryId) 2892567029667191230-7DA6C0A855122CF5-190
Fileset gpfs1022 changed.
Promoted fileset gpfs1022 to Primary
```

Acting primary:

```
/usr/lpp/mmfs/bin/mmlsfileset homefs1 gpfs1022 -L -afm

Filesets in file system 'homefs1':

Attributes for fileset gpfs1022:
=================================
Status                                  Linked
Path                                    /gpfs/homefs1/gpfs1022
Id                                      190
Root inode                              1345847299
Parent Id                               0
Created                                 Tue Apr 28 03:53:03 2015
Comment
Inode space                             87
Maximum number of inodes                100032
Allocated inodes                        100032
Permission change flag                  chmodAndSetacl
afm-associated                          Yes
Target                                  nfs://(null)(null)
Mode                                    primary
Async Delay                             15 (default)
Recovery Point Objective                disable
Last pSnapId                            0
Number of Gateway Flush Threads         4
Primary Id                              2892567029667191230-7DA6C0A855122CF5-190
```

Contents from acting primary after failover:

```
ls -l /gpfs/homefs1/gpfs1022
total 614400
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_4
```

8. Create data from acting primary:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /gpfs/homefs1/gpfs1022/file_actingpri_1
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 52699.54 Kbytes/sec, thread utilization 0.991

/usr/lpp/mmfs/samples/perf/gpfsperf create seq /gpfs/homefs1/gpfs1022/file_actingpri_2
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 56026.22 Kbytes/sec, thread utilization 0.999
```

Data from acting primary:

```
ls -l /gpfs/homefs1/gpfs1022/
total 815104
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_4
```

9. New primary created. Prepare a snapshot for new primary setup from acting primary:

```
/usr/lpp/mmfs/bin/mmafmctl homefs1 replacePrimary -j gpfs1022
psnap0-newprimary-base-rpo-7DA6C0A855122CF5-190
```

10. Create new GPFS fileset at new primary site:

```
mmcrfileset fs1 newgpfs1022 --inode-space=new
Fileset newgpfs1022 created with id 187 root inode 49807363.

mmlinkfileset fs1 newgpfs1022 -J /gpfs/fs1/newgpfs1022
Fileset newgpfs1022 linked at /gpfs/fs1/newgpfs1022
```

11. Copy/Truck contents from snapshot created in step 9, to new primary outband through **scp**:

```
psnap0-newprimary-base-rpo-7DA6C0A855122CF5-190
```

New primary contents:

```
ls -l /gpfs/fs1/newgpfs1022
total 1222656
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_4
```

Create more data from acting primary in the meanwhile, to show that applications continue there:

```
/usr/lpp/mmfs/samples/perf/gpfsperf create seq /gpfs/homefs1/gpfs1022/file_actingpri_3
  recSize 10K nBytes 100M fileSize 100M
  nProcesses 1 nThreadsPerProcess 1
  file cache flushed before test
  not using direct I/O
  offsets accessed will cycle through the same file segment
  not using shared memory buffer
  not releasing byte-range token after open
  no fsync at end of test
    Data rate was 53044.68 Kbytes/sec, thread utilization 0.987
```

12. Convert new GPFS fileset to primary using **--outband** option:

```
/usr/lpp/mmfs/bin/mmafmctl fs1 convertToPrimary -j newgpfs1022 --afmtarget gpfs:///gpfs/remotefs1/gpfs1022
   --secondary-snapname psnap0-newprimary-base-rpo-7DA6C0A855122CF5-190

Checking for any special files. This may take a while...
Converting GPFS fileset to AFM primary fileset...
Primary Id (afmPrimaryId) 3235816757204815584-C0A802135530BD89-187
Fileset newgpfs1022 changed.
Flushing dirty data for snapshot newgpfs1022::psnap0-newprimary-base-rpo-7DA6C0A855122CF5-190...
Quiescing all file system operations.
Snapshot newgpfs1022::psnap0-newprimary-base-rpo-7DA6C0A855122CF5-190 created with id 332.
```

New primary contents:

```
ls -l /gpfs/fs1/newgpfs1022
total 1228800
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_4
```

13. Run failback start on new primary:

```
/usr/lpp/mmfs/bin/mmafmctl fs1 failbackToPrimary -j newgpfs1022 --start
Fileset newgpfs1022 changed.
mmafmctl: failbackToPrimary restoring from psnap psnap0-newprimary-base-rpo-7DA6C0A855122CF5-190
[2015-04-28 03:55:31] Restoring fileset "newgpfs1022" from snapshot "psnap0-newprimary-base-rpo-7DA6C0A855122CF5-190" of filesystem "/dev/fs1"
[2015-04-28 03:55:33] Scanning inodes, phase 1 ...
[2015-04-28 03:55:33] 49907392 inodes have been scanned, 100% of total.
[2015-04-28 03:55:33] There's no data changes since the restoring snapshot, skipping restore.
[2015-04-28 03:55:33] Restore completed successfully.
[2015-04-28 03:55:33] Clean up.
```

Run **applyUpdates** to sync up primary to acting primary:

```
/usr/lpp/mmfs/bin/mmafmctl fs1 applyUpdates -j newgpfs1022
[2015-04-28 03:55:36] Getting the list of updates from the acting primary...
[2015-04-28 03:55:45] Applying the 3 updates...
[2015-04-28 03:55:47] 3 updates have been applied, 100% of total.
Flushing dirty data for snapshot newgpfs1022::failback-psnap-rpo-C0A802135530BD89-187-15-04-28-03-55-35...
Quiescing all file system operations.
Snapshot newgpfs1022::failback-psnap-rpo-C0A802135530BD89-187-15-04-28-03-55-35 created with id 333.
```

Complete failback process on new primary:

```
/usr/lpp/mmfs/bin/mmafmctl fs1 failbackToPrimary -j newgpfs1022 --stop
Fileset newgpfs1022 changed.
```

14. Convert the acting primary back to secondary and re-establish the relationship:

```
/usr/lpp/mmfs/bin/mmchfileset homefs1 gpfs1022 -p
 afmmode=secondary,afmPrimaryId=3235816757204815584-C0A802135530BD89-187
Fileset gpfs1022 changed.
```

Primary contents:

```
ls -l /gpfs/fs1/newgpfs1022
total 1433600
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_gpfspri_4
```

Secondary contents:

```
ls -l /gpfs/homefs1/gpfs1022
total 921600
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_actingpri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_actingpri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:55 file_actingpri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_1
-rw-r--r-- 1 root root 104857600 Apr 28 03:53 file_gpfspri_2
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_3
-rw-r--r-- 1 root root 104857600 Apr 28 03:54 file_gpfspri_4
```

Chapter 16. Protocols cluster disaster recovery

Protocols cluster disaster recovery (DR) uses the capabilities of Active File Management (AFM) based Async Disaster Recovery (AFM DR) features to provide a solution that allows an IBM Spectrum Scale cluster to fail over to another cluster and fail back, and backup and restore the protocol configuration information in cases where a secondary cluster is not available.

| For more information on AFM-based Async DR, see the topic *AFM-based Async Disaster Recovery* in the
| *IBM Spectrum Scale: Advanced Administration Guide*.

Although an overview of the steps that need to be done is provided if performing these operations manually, it is recommended to use the **mmcesdr** command because it automates DR setup, failover, failback, backup, and restore actions. For more information about the **mmcesdr** command, see *mmcesdr command* in *IBM Spectrum Scale: Administration and Programming Reference* .

Protocols cluster disaster recovery limitations and prerequisites

For protocols cluster disaster recovery (DR) in an IBM Spectrum Scale cluster, the prerequisites and limitations are as follows.

Ensure that the following prerequisites are met for the secondary cluster for disaster recovery in an IBM Spectrum Scale with protocols.

- IBM Spectrum Scale is installed and configured.
- Cluster Export Services (CES) are installed and configured, and the shared root file system is defined.
- All protocols that are configured on the primary cluster are also configured on the secondary cluster.
- Authentication on secondary cluster is identical to the authentication on the primary cluster.
- All exports that need to be protected using AFM DR must have the same device and fileset name, and the same fileset link point on the secondary cluster as defined on the primary cluster.
- IBM NFSv3 stack must be configured on home cluster for the AFM DR transport of data.
- No data must be written to exports on secondary cluster while cluster is acting only as a secondary cluster, before a failover.

The following limitations apply for disaster recovery in an IBM Spectrum Scale cluster with protocols.

- Only data contained within independent filesets can be configured for AFM based Async Disaster Recovery (AFM DR). Therefore, all protocol exports that you want to be protected by DR must have the export path equal to the independent fileset link point.
- Nested independent or dependent filesets are not supported.
- Backup and restore of the authentication configuration is not supported.
- On failover and failback or restore, all clients need to disconnect and then reconnect.
- If **--file-config --restore** is specified, perform the follow steps:
  - On failover: file authentication must be removed and then reconfigured on the secondary cluster.
  - On restore: file authentication must be removed and then reconfigured on the primary cluster.
  - On failback: file authentication must be removed and then reconfigured on both primary and secondary clusters.
- Multi-region object deployment is not supported with protocols cluster DR. Therefore, if multi-region object deployment is enabled, object data or configuration information is not protected through protocols cluster DR.

- Tivoli Storage Manager for Space Management and LTFS migrated data within protocol exports is not supported within protocols cluster DR.
- Tivoli Storage Manager configuration file information is not automatically protected through protocols cluster DR.

Example setup for protocols disaster recovery

The following example scenario is used to show how to set up disaster recovery functionality for an IBM Spectrum Scale cluster with protocols.

This example consists of three NFS exports, three SMB exports, one object fileset, and two unified file and object access filesets that are also NFS exports. For the NFS and SMB exports, only two of each are independent filesets. This allows an AFM-based Async DR (AFM DR) configuration. For simplification, the filesets are named according to whether or not they were dependent or independent for the NFS and SMB exports. The inclusion of dependent filesets as exports is to show the warnings that are given when an export path is not an independent fileset link point.

Note: NFS and SMB exports must be named according to their fileset link point names for them to be captured by the **mmcesdr** command for protocols cluster disaster recovery. For example, if you have a fileset `nfs-smb-combo`, the NFS or the SMB export name must be *GPFS_Path*/`nfs-smb-combo`. If you use a name in the fileset's subdirectory for the NFS or the SMB export (for example: *GPFS_Path*/`nfs-smb-combo/nfs1`), the **mmcesdr** command does not capture that export.

NFS exports
- `/gpfs/fs0/nfs-ganesha-dep`
- `/gpfs/fs0/nfs-ganesha1`
- `/gpfs/fs0/nfs-ganesha2`

SMB exports
- `/gpfs/fs0/smb1`
- `/gpfs/fs0/smb2`
- `/gpfs/fs0/smb-dep`

Combination NFS and SMB exports
- `/gpfs/fs0/combo1`
- `/gpfs/fs0/combo2`

Object fileset
- `/gpfs/fs1/object_fileset`

Unified file and object access filesets
- `/gpfs/fs1/obj_sofpolicy1`

  This fileset is created by creating a storage policy using the **`mmobj policy create sofpolicy1 --enable-file-access`** command.
- `/gpfs/fs1/obj_sofpolicy2`

  This fileset is created by creating a storage policy using the **`mmobj policy create sofpolicy2 --enable-file-access --enable-compression --compression-schedule "30:02:*:*"`** command.

Setting up gateway nodes to ensure cluster communication during failover

Both the primary and the DR clusters require designating gateway nodes for access to whichever side is acting as the cache. By designating gateway nodes on both clusters, you can ensure that even during failover, cluster communication continues properly.

To handle a possible node failure, you need to specify at least two nodes on each cluster to be gateway nodes. To specify two nodes on the primary cluster as gateway nodes, use the command similar to the following:

mmchnode -N Node1,Node2 --gateway

Using the example setup mentioned in “Example setup for protocols disaster recovery” on page 262, the command to specify gateway nodes on the primary cluster is as follows:

```
# mmchnode -N zippleback-vm1,zippleback-vm2 --gateway
Tue Apr 28 20:59:01 MST 2015: mmchnode: Processing node zippleback-vm2
Tue Apr 28 20:59:01 MST 2015: mmchnode: Processing node zippleback-vm1
mmchnode: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
# Tue Apr 28 20:59:04 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation started

Tue Apr 28 20:59:08 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation completed; mmdsh rc=0
```

Similarly, you need to specify at least two nodes on the DR cluster as gateway nodes. Using the example setup, the command to specify gateway nodes on the DR cluster is as follows:

```
# mmchnode -N windwalker-vm1,windwalker-vm2 --gateway
Tue Apr 28 20:59:49 MST 2015: mmchnode: Processing node windwalker-vm2
Tue Apr 28 20:59:49 MST 2015: mmchnode: Processing node windwalker-vm1
mmchnode: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
# Tue Apr 28 20:59:51 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation started

Tue Apr 28 20:59:54 MST 2015: mmcommon pushSdr_async:
mmsdrfs propagation completed; mmdsh rc=0
```

Creating the inband disaster recovery setup

Use the following steps for inband disaster recovery setup in an IBM Spectrum Scale cluster with protocols.

1. On the primary cluster, use the following command to configure independent fileset exports as AFM DR filesets and back up configuration information.

   ```
   mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo 10 --inband
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/5, configuration fileset creation/verification.
   Successfully completed step 1/5, configuration fileset creation/verification.
   Performing step 2/5, protocol and export services configuration backup.
   Successfully completed step 2/5, protocol and export services configuration backup.
   Performing step 3/5, determination of protocol exports to protect with AFM DR.
   WARNING: Export /gpfs/fs0/nfs-ganesha-dep of type nfs will NOT be protected
   through AFM DR because it is a dependent fileset.
   Not all exports of type NFS-ganesha will be protected through AFM DR, rc: 2
   WARNING: Export /gpfs/fs0/smb-dep of type smb will NOT be protected
   through AFM DR because it is a dependent fileset.
   Not all exports of type SMB will be protected through AFM DR, rc: 2
   Completed with errors step 3/5, determination of protocol exports to protect with AFM DR.
   Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
   ```

```
Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
Performing step 5/5, creation of output DR configuration file.
Successfully completed step 5/5, creation of output DR configuration file.

File to be used with secondary cluster in next step of cluster DR setup: /root//DR_Config
```

Note: In this command example, there are two exports that are not protected. During the configuration step, any exports that are not protected through AFM DR generate a warning to the standard output of the command.

2. Use the following command to transfer the DR configuration file from the primary cluster to the secondary cluster.

   ```
   scp /root//DR_Config windwalker-vm1:/root/
   ```

   The system displays output similar to the following:

   ```
   root@windwalker-vm1's password:
   DR_Config 100% 1551 1.5KB/s 00:00
   ```

3. On the secondary cluster, use the following command to create the independent filesets that will be a part of the pair of AFM DR filesets associated with those on the primary cluster. In addition to creating filesets, this command also creates the necessary NFS exports.

   ```
   mmcesdr secondary config --input-file-path /root/ --inband
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/3, creation of independent filesets to be used for AFM DR.
   Successfully completed step 1/3, creation of independent filesets to be used for AFM DR.
   Performing step 2/3, creation of NFS exports to be used for AFM DR.
   Successfully completed step 2/3, creation of NFS exports to be used for AFM DR.
   Performing step 3/3, conversion of independent filesets to AFM DR secondary filesets.
   Successfully completed step 3/3, conversion of independent filesets to AFM DR secondary filesets.
   ```

4. Ensure that all of the expected AFM DR pairs show as `Active` in the output of the **`mmafmctl`** command and take corrective action if they do not.

   ```
   # mmafmctl fs0 getstate

   Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
   ------------ -------------- ------------- ------------ ------------ -------------
   nfs-ganesha1 nfs://9.11.102.210/gpfs/fs0/nfs-ganesha1 Active zippleback-vm2 0 4
   nfs-ganesha2 nfs://9.11.102.211/gpfs/fs0/nfs-ganesha2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4
   combo1 nfs://9.11.102.210/gpfs/fs0/combo1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 7
   combo2 nfs://9.11.102.211/gpfs/fs0/combo2 Active zippleback-vm2 0 66
   smb1 nfs://9.11.102.210/gpfs/fs0/smb1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 65
   smb2 nfs://9.11.102.211/gpfs/fs0/smb2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4

   # mmafmctl fs1 getstate

   Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
   ------------ -------------- ------------- ------------ ------------ -------------
   object_fileset nfs://9.11.102.211/gpfs/fs1/object_fileset Active zippleback-vm1.tuc.stglabs.ibm.com 0 95671
   obj_sofpolicy1 nfs://9.11.102.211/gpfs/fs1/obj_sofpolicy1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 27
   obj_sofpolicy2 nfs://9.11.102.210/gpfs/fs1/obj_sofpolicy2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 26
   async_dr nfs://9.11.102.210/gpfs/fs1/.async_dr Active zippleback-vm1.tuc.stglabs.ibm.com 0 2751
   ```

   Note: A state of `Dirty` is normal when data is actively being transferred from the primary cluster to the secondary cluster.

Alternate Inband Set up showing the `--allowed-nfs-clients` parameter being used

With the addition of the new optional parameter **`--allowed-nfs-clients`** you can specify exactly which clients are allowed to connect to the NFS transport exports that are created on the secondary cluster. This parameter can be used for both inband and outband set up. Here is an example of the parameter being used for an inband setup:

- On the primary cluster, run the following command to configure independent fileset exports as AFM DR filesets and to backup configuration information:
  `mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo 10 --inband --allowed-nfs-clients --gateway-nodes`

This command will give the following output:

```
Performing step 1/5, configuration fileset creation/verification.
Successfully completed step 1/5, configuration fileset creation/verification.
Performing step 2/5, protocol and export services configuration backup.
Successfully completed step 2/5, protocol and export services configuration backup.
Performing step 3/5, determination of protocol exports to protect with AFM DR.
WARNING: Export /gpfs/fs0/nfs-ganesha-dep of type nfs will NOT be protected through AFM DR because it is a dependent fileset.
Not all exports of type NFS-ganesha will be protected through AFM DR, rc: 2
WARNING: Export /gpfs/fs0/smb-dep of type smb will NOT be protected through AFM DR because it is a dependent fileset.
Not all exports of type SMB will be protected through AFM DR, rc: 2
Completed with errors step 3/5, determination of protocol exports to protect with AFM DR.
Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
Performing step 5/5, creation of output DR configuration file.
Successfully completed step 5/5, creation of output DR configuration file.
```

Creating the outband disaster recovery setup

Use the following steps for outband disaster recovery setup in an IBM Spectrum Scale cluster with protocols.

1. On the primary cluster, use the following command to configure independent fileset exports as AFM DR filesets and back up configuration information.

   ```
   mmcesdr primary config --output-file-path /root/ --ip-list "9.11.102.211,9.11.102.210" --rpo 10
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/5, configuration fileset creation/verification.
   Successfully completed step 1/5, configuration fileset creation/verification.
   Performing step 2/5, protocol and export services configuration backup.
   Successfully completed step 2/5, protocol and export services configuration backup.
   Performing step 3/5, determination of protocol exports to protect with AFM DR.
   Successfully completed step 3/5, determination of protocol exports to protect with AFM DR.
   Performing step 4/5, conversion of protected filesets into AFM DR primary filesets.
   Successfully completed step 4/5, conversion of protected filesets into AFM DR primary filesets.
   Performing step 5/5, creation of output DR configuration file.
   Successfully completed step 5/5, creation of output DR configuration file.

   File to be used with secondary cluster in next step of cluster DR setup: /root//DR_Config
   ```

2. Use the following command to transfer the DR configuration file from the primary cluster to the secondary cluster.

   ```
   scp /root//DR_Config windwalker-vm1:/root/
   ```

   The system displays output similar to the following:

   ```
   root@windwalker-vm1's password:
   DR_Config 100% 2566 2.5KB/s 00:00
   ```

3. On the secondary cluster, use the following command to create the independent filesets that will later be paired with those on the primary cluster to form AFM DR pairs.

   ```
   mmcesdr secondary config --input-file-path /root --prep-outband-transfer
   ```

   The system displays output similar to the following:

   ```
   Creating independent filesets to be used as recipients of AFM DR outband transfer of data.
   Transfer all data on primary cluster for fileset fs0:combo1 to fileset fs0:combo1 on secondary
   cluster.
   Transfer all data on primary cluster for fileset fs0:combo2 to fileset fs0:combo2 on secondary
   cluster.
   Transfer all data on primary cluster for fileset fs0:nfs-ganesha1 to fileset fs0:nfs-ganesha1 on
   secondary cluster.
   Transfer all data on primary cluster for fileset fs0:nfs-ganesha2 to fileset fs0:nfs-ganesha2 on
   secondary cluster.
   Transfer all data on primary cluster for fileset fs0:smb1 to fileset fs0:smb1 on secondary cluster.
   Transfer all data on primary cluster for fileset fs0:smb2 to fileset fs0:smb2 on secondary cluster.
   Transfer all data on primary cluster for fileset fs1:async_dr to fileset fs1:async_dr on secondary
   cluster.
   Transfer all data on primary cluster for fileset fs1:obj_sofpolicy1 to fileset fs1:obj_sofpolicy1 on
   ```

```
secondary cluster.

Transfer all data on primary cluster for fileset fs1:obj_sofpolicy2 to fileset fs1:obj_sofpolicy2 on
secondary cluster.

Transfer all data on primary cluster for fileset fs1:object_fileset to fileset fs1:object_fileset on
secondary cluster.

Successfully completed creating independent filesets to be used as recipients of AFM DR outband
transfer of data.
Transfer data from primary cluster through outbound trucking to the newly created independent filesets
before proceeding to the next step.
```

4. Transfer the data from all protected filesets on the primary cluster to the corresponding filesets on the secondary cluster. If transferring files for which GPFS extended attributes also need to be transferred (such as object data), you must use a method that transfers GPFS extended attributes. For this purpose, you can use IBM Spectrum Protect (to back up data to tape and then restore it), GPFS cross-cluster mount, and AFM. Standard SCP and standard rsync do not transfer GPFS extended attributes.
5. After all the data has been transferred to the secondary cluster, use the following command to complete the setup on the secondary cluster.

   ```
   mmcesdr secondary config --input-file-path /root
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/3, verification of independent filesets to be used for AFM DR.
   Successfully completed 1/3, verification of independent filesets to be used for AFM DR.
   Performing step 2/3, creation of NFS exports to be used for AFM DR.
   Successfully completed step 2/3, creation of NFS exports to be used for AFM DR.
   Performing step 3/3, conversion of independent filesets to AFM DR secondary filesets.
   Successfully completed step 3/3, conversion of independent filesets to AFM DR secondary filesets.
   ```

6. Ensure that all of the expected AFM DR pairs show as `Active` in the output of the **mmafmctl** command and take corrective action if they do not.

   ```
   # mmafmctl fs0 getstate

   Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
   ------------ -------------- ------------- ------------ ------------ -------------
   nfs-ganesha1 nfs://9.11.102.210/gpfs/fs0/nfs-ganesha1 Active zippleback-vm2 0 4
   nfs-ganesha2 nfs://9.11.102.211/gpfs/fs0/nfs-ganesha2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4
   combo1 nfs://9.11.102.210/gpfs/fs0/combo1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 7
   combo2 nfs://9.11.102.211/gpfs/fs0/combo2 Active zippleback-vm2 0 66
   smb1 nfs://9.11.102.210/gpfs/fs0/smb1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 65
   smb2 nfs://9.11.102.211/gpfs/fs0/smb2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 4

   # mmafmctl fs1 getstate

   Fileset Name Fileset Target Cache State Gateway Node Queue Length Queue numExec
   ------------ -------------- ------------- ------------ ------------ -------------
   object_fileset nfs://9.11.102.211/gpfs/fs1/object_fileset Active zippleback-vm1.tuc.stglabs.ibm.com 0 95671
   obj_sofpolicy1 nfs://9.11.102.211/gpfs/fs1/obj_sofpolicy1 Active zippleback-vm1.tuc.stglabs.ibm.com 0 27
   obj_sofpolicy2 nfs://9.11.102.210/gpfs/fs1/obj_sofpolicy2 Active zippleback-vm1.tuc.stglabs.ibm.com 0 26
   async_dr nfs://9.11.102.210/gpfs/fs1/.async_dr Active zippleback-vm1.tuc.stglabs.ibm.com 0 2751
   ```

   Note: A state of `Dirty` is normal when data is actively being transferred from the primary cluster to the secondary cluster.

Performing failover when primary cluster fails

The failover procedure can use a new option to choose whether or not file protocols have their shares re-created or if the entire file protocol configuration for NFS and SMB is restored. The default is to re-create the NFS and SMB shares because it does not require removing and adding the file authentication again afterwards. The failover can be performed in one of the following ways:

Re-create file export configuration

Use the following steps on the secondary cluster to fail over when the primary cluster fails in an IBM Spectrum Scale cluster with protocols.

On the secondary cluster, after the primary cluster has failed, use the following command.

```
mmcesdr secondary failover
```

The system displays output similar to the following:

```
Performing step 1/4, saving current NFS configuration to restore after failback.
Successfully completed step 1/4, saving current NFS configuration to restore after failback.
Performing step 2/4, failover of secondary filesets to primary filesets.
Successfully completed step 2/4, failover of secondary filesets to primary filesets.
Performing step 3/4, protocol configuration restore.
Successfully completed step 3/4, protocol configuration restore.
Performing step 4/4, create/verify NFS AFM DR transport exports.
Successfully completed step 4/4, create/verify NFS AFM DR transport exports.
```

Restore file export configuration

Use the following steps on the secondary cluster to fail over when the primary cluster fails in an IBM Spectrum Scale cluster with protocols.

1. On the secondary cluster, after the primary cluster has failed, use the following command.

   ```
   mmcesdr secondary failover --file-config --restore
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/4, saving current NFS configuration to restore after failback.
   Successfully completed step 1/4, saving current NFS configuration to restore after failback.
   Performing step 2/4, failover of secondary filesets to primary filesets.
   Successfully completed step 2/4, failover of secondary filesets to primary filesets.
   Performing step 3/4, protocol configuration/exports restore.
   Successfully completed step 3/4, protocol configuration/exports restore.
   Performing step 4/4, create/verify NFS AFM DR transport exports.
   Successfully completed step 4/4, create/verify NFS AFM DR transport exports.

   ================================================================================
   = If all steps completed successfully, please remove and then re-create file
   = authentication on the DR cluster.
   = Once this is complete, Protocol Cluster Failover will be complete.
   ================================================================================
   ```

2. Remove the file authentication on the secondary cluster after failover and then add back the file authentication before failover is considered to be complete and client operations can resume, but point to the secondary cluster.

Performing failback to old primary

When failing back to the old primary, the file protocol configuration can either be re-created or restored on the old primary. The NFS transport exports on the secondary need to be re-created or NFS configuration can be restored.

Re-create file protocol configuration for old primary

In the following example, file protocol configuration is re-created. To re-create the file exports on the old primary during restore, one of these commands can be run: **`mmcesdr primary restore`** or **`mmcesdr primary restore --file-config --recreate`**. The completion of failback on the secondary where the NFS transport export is re-created can also be performed by running one of these commands: **`mmcesdr secondary failback --post-failback-complete`** or **`mmcesdr secondary failback --post-failback-complete --file-config --recreate`**. Use the following steps for failing back to an old primary cluster in an IBM Spectrum Scale cluster with protocols.

1. On the old primary cluster, use the following command.

```
mmcesdr primary failback --start --input-file-path "/root/"
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

The system displays output similar to the following:

```
Performing failback to primary on all AFM DR protected filesets.
Successfully completed failback to primary on all AFM DR protected filesets
```

2. On the old primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 25 Mins. 20 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before stopping
   failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

3. On the secondary cluster (acting primary), quiesce all client operations.
4. On the old primary cluster, use the following command one more time.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 0 Mins. 27 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before stopping
   failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

5. On the old primary cluster, use the following command.

   ```
   mmcesdr primary failback --stop --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing stop of failback to primary on all AFM DR protected filesets.
   Successfully completed stop failback to primary on all AFM DR protected filesets.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

6. On the old primary cluster, use the following command to restore configuration.

   ```
   mmcesdr primary restore
   ```

   The system displays output similar to the following:

   ```
   Restoring cluster and enabled protocol configurations/exports.
   Successfully completed restoring cluster and enabled protocol configurations/exports.
   ```

7. On the secondary cluster (acting primary), use the following command to convert it back to a secondary cluster and associate it with the original primary cluster.

   ```
   mmcesdr secondary failback --post-failback-complete
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
   Successfully completed step 1/2, converting protected filesets back into AFM DR secondary
   filesets.
   Performing step 2/2, restoring/recreating AFM DR-based NFS share configuration.
   Successfully completed step 2/2, restoring/recreating AFM DR-based NFS share configuration.
   ```

```
================================================================================
= If all steps completed successfully, remove and then re-create file
= authentication on the Secondary cluster.
= Once this is complete, Protocol Cluster Failback will be complete.
================================================================================
```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

Restore file protocol configuration for old primary

In the following example, file protocol configuration is restored. To restore the file exports on the old primary during restore, run the following command: **mmcesdr primary restore --file-config --restore**. The completion of failback on the secondary where the NFS transport export is re-created can also be performed by running this commands: **mmcesdr secondary failback --post-failback-complete --file-config --restore**. Use the following steps for failing back to an old primary cluster in an IBM Spectrum Scale cluster with protocols.

1. On the old primary cluster, use the following command.

   ```
   mmcesdr primary failback --start --input-file-path "/root/"
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

   The system displays output similar to the following:

   ```
   Performing failback to primary on all AFM DR protected filesets.
   Successfully completed failback to primary on all AFM DR protected filesets
   ```

2. On the old primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 25 Mins. 20 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before stopping
   failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

3. On the secondary cluster (acting primary), quiesce all client operations.
4. On the old primary cluster, use the following command one more time.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:object_fileset and is 0 Hrs. 0 Mins. 27 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before stopping
   failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

5. On the old primary cluster, use the following command.

   ```
   mmcesdr primary failback --stop --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing stop of failback to primary on all AFM DR protected filesets.
   Successfully completed stop failback to primary on all AFM DR protected filesets.
   ```

Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

6. On the old primary cluster, use the following command to restore configuration:

   ```
   mmcesdr primary restore --file-config --restore
   ```

   The system displays output similar to the following:

   ```
   Restoring cluster and enabled protocol configurations/exports.
   Successfully completed restoring cluster and enabled protocol configurations/exports.

   ================================================================================
   =  If all steps completed successfully, remove and then re-create file
   =  authentication on the Primary cluster.
   =  Once this is complete, Protocol Cluster Configuration Restore will be complete.
   ================================================================================
   ```

7. On the primary cluster, remove the file authentication and then add it again.
8. On the secondary cluster (acting primary), use the following command to convert it back to a secondary cluster and associate it with the original primary cluster.

   ```
   mmcesdr secondary failback --post-failback-complete --input-file-path /root --file-config --restore
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
   Successfully completed step 1/2, converting protected filesets back into AFM DR secondary
   filesets.
   Successfully completed step 1/2, converting protected filesets back into AFM DR secondary filesets.
   Performing step 2/2, restoring/recreating AFM DR-based NFS share configuration.
   Successfully completed step 2/2, restoring/recreating AFM DR-based NFS share configuration.

   ================================================================================
   = If all steps completed successfully, remove and then re-create file
   = authentication on the Secondary cluster.
   = Once this is complete, Protocol Cluster Failback will be complete.
   ================================================================================
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

9. On the secondary cluster, remove the file authentication and then add it again.

Performing failback to new primary

When failing back to the new primary, file protocol configuration can either be re-created or restored on the new primary. The NFS transport exports on the secondary need to be re-created or NFS configuration can be restored.

Re-create file protocol configuration for new primary

The file protocol configuration is re-created in the following example. To re-create the file exports on the new primary during restore, one of these commands can be run:
mmcesdr primary restore or **mmcesdr primary restore --file-config --recreate**.
The completion of failback on the secondary where the NFS transport exports is re-created can also be performed by running one of these commands:
mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root" or
mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root" --file-config --recreate.
Use the following steps for failing over to a new primary cluster in an IBM Spectrum Scale cluster with protocols.

1. On the old secondary cluster, use the following command to prepare recovery snapshots that contain data that will be transferred to the new primary cluster.

   ```
   mmcesdr secondary failback --generate-recovery-snapshots --output-file-path "/root/"
   --input-file-path "/root/"
   ```

The system displays output similar to the following:

```
Performing step 1/2, generating recovery snapshots for all AFM DR acting primary filesets.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/combo1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-1 to fileset link point of
fileset fs0:combo1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/combo2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-2 to fileset link point of
fileset fs0:combo2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/nfs-ganesha1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-3 to fileset link
point of fileset fs0:nfs-ganesha1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/nfs-ganesha2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-4 to fileset link
point of fileset fs0:nfs-ganesha2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/smb1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-5 to fileset link point of
fileset fs0:smb1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6 to fileset link point of
fileset
fs0:smb2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/.async_dr/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-2 to fileset link
point of fileset fs1:async_dr on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-3 to fileset link
point of fileset fs1:obj_sofpolicy1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-4 to fileset link
point of fileset fs1:obj_sofpolicy2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/object_fileset/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-1 to fileset link
point of fileset fs1:object_fileset on new primary cluster.
Successfully completed step 1/2, generating recovery snapshots for all AFM DR acting primary
filesets.
Performing step 2/2, creation of recovery output file for failback to new primary.
Successfully completed step 2/2, creation of recovery output file for failback to new primary.

File to be used with new primary cluster in next step of failback to new primary cluster:
/root//DR_Config
```

2. Transfer the newly created DR configuration file to the new primary cluster.

   ```
   scp /root//DR_Config zippleback-vm1:/root/
   ```

   The system displays output similar to the following:

   ```
   root@zippleback-vm1's password:
   DR_Config 100% 1996 2.0KB/s 00:00
   ```

3. On the new primary cluster, use the following command to create the independent filesets that will receive the data transferred from the recovery snapshots.

   ```
   mmcesdr primary failback --prep-outband-transfer --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Creating independent filesets to be used as recipients of AFM DR outband transfer of data.
   Successfully completed creating independent filesets to be used as recipients of AFM DR outband
   transfer of data.
   ```

Transfer data from recovery snapshots through outbound trucking to the newly created independent filesets before proceeding to the next step.

4. Transfer data from within the recovery snapshots of the secondary cluster to the new primary cluster.

   Note: Only one transfer is shown in the example below.

   ```
   rsync -av /gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6/*
   zippleback-vm1:/gpfs/fs0/smb2/
   ```

The system displays output similar to the following:

```
root@zippleback-vm1's password:
sending incremental file list
test

sent 68 bytes received 31 bytes 15.23 bytes/sec
total size is 0 speedup is 0.00
```

Attention: When transferring files that need to also transfer GPFS extended attributes, extra steps are required. This example uses standard rsync which does not transfer extended attributes.

5. On the new primary cluster, use the following command to convert the independent filesets to primary filesets and generate a new DR configuration file that will be used on the primary cluster for the next steps and then transferred to the secondary cluster to be used in a later step.

   ```
   mmcesdr primary failback --convert-new --output-file-path /root/ --input-file-path /root/
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/2, conversion of independent filesets into new primary filesets to be used for AFM DR.
   Successfully completed step 1/2, failback to primary on all AFM DR protected filesets.
   Performing step 2/2, creation of output file for remaining failback to new primary steps.
   Successfully completed step 2/2, creation of output file for remaining failback to new primary steps.

   File to be used with new primary cluster in next step of failback to new primary cluster: /root//DR_Config
   ```

6. On the new primary cluster, use the following command.

   ```
   mmcesdr primary failback --start --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing failback to primary on all AFM DR protected filesets.
   Successfully completed failback to primary on all AFM DR protected filesets.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

7. On the new primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 45 Mins. 10 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```

8. On the secondary cluster (acting primary), quiesce all client operations.

9. On the new primary cluster, use the following command one more time.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 0 Mins. 16 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.

10. On the new primary cluster, use the following command to stop the failback process and convert the new primary filesets to read/write.

    ```
    mmcesdr primary failback --stop --input-file-path "/root/"
    ```

    The system displays output similar to the following:

```
Performing stop of failback to primary on all AFM DR protected filesets.
Successfully completed stop failback to primary on all AFM DR protected filesets.
```

11. On the new primary cluster, use the following command to restore the protocol and export services configuration information.

    ```
    mmcesdr primary restore --new-primary
    ```

    Note: The `--new-primary` option must be used to ensure protocol configuration is restored correctly.

    The system displays output similar to the following:

    ```
    Restoring cluster and enabled protocol
              configurations/exports.Successfully completed restoring cluster and enabled protocol
              configurations/exports.
    ```

12. Transfer the updated DR configuration file from the new primary cluster to the secondary cluster.

    ```
    scp /root//DR_Config windwalker-vm1:/root/
    ```

    The system displays output similar to the following:

    ```
    root@windwalker-vm1's password:
    DR_Config 100% 2566 2.5KB/s 00:00
    ```

13. On the secondary cluster, use the following command to register the new primary AFM IDs to the independent filesets on the secondary cluster acting as part of the AFM DR pairs.

    ```
    mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root"
    ```

    The system displays output similar to the following:

    ```
    Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
    Successfully completed step 1/2, converting protected filesets back into AFM DR secondary filesets.
    Performing step 2/2, recreating AFM DR-based NFS share configuration.
    Successfully completed step 2/2, recreating AFM DR-based NFS share configuration.

    ================================================================================
    = If all steps completed successfully, remove and then re-create file
    = authentication on the Secondary cluster.
    = Once this is complete, Protocol Cluster Failback will be complete.
    ================================================================================
    ```

Restore file protocol configuration for new primary

The file protocol configuration is restored in the following example. To restore the file exports on the old primary during restore, run the following command: **`mmcesdr primary restore --file-config --restore`**. The completion of failback on the secondary where the NFS transport export is re-created can also be performed by running this commands: **`mmcesdr secondary failback --post-failback-complete --file-config --restore`**. Use the following steps for failing back to an old primary cluster in an IBM Spectrum Scale cluster with protocols

1. On the old secondary cluster, use the following command to prepare recovery snapshots that contain data that will be transferred to the new primary cluster.

    ```
    mmcesdr secondary failback --generate-recovery-snapshots --output-file-path "/root/"
    --input-file-path "/root/"
    ```

    The system displays output similar to the following:

    ```
    Performing step 1/2, generating recovery snapshots for all AFM DR acting primary filesets.
    Transfer all data under snapshot located on acting primary cluster at:
    /gpfs/fs0/combo1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-1 to fileset link point of
    fileset fs0:combo1 on new primary cluster.
    Transfer all data under snapshot located on acting primary cluster at:
    /gpfs/fs0/combo2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-2 to fileset link point of
    fileset fs0:combo2 on new primary cluster.
    Transfer all data under snapshot located on acting primary cluster at:
    /gpfs/fs0/nfs-ganesha1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-3 to fileset link
    point of fileset fs0:nfs-ganesha1 on new primary cluster.
    Transfer all data under snapshot located on acting primary cluster at:
    /gpfs/fs0/nfs-ganesha2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-4 to fileset link
    point of fileset fs0:nfs-ganesha2 on new primary cluster.
    Transfer all data under snapshot located on acting primary cluster at:
    ```

```
/gpfs/fs0/smb1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-5 to fileset link point of
fileset fs0:smb1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6 to fileset link point of
fileset
fs0:smb2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/.async_dr/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-2 to fileset link
point of fileset fs1:async_dr on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy1/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-3 to fileset link
point of fileset fs1:obj_sofpolicy1 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/obj_sofpolicy2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-4 to fileset link
point of fileset fs1:obj_sofpolicy2 on new primary cluster.
Transfer all data under snapshot located on acting primary cluster at:
/gpfs/fs1/object_fileset/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DECB-1 to fileset link
point of fileset fs1:object_fileset on new primary cluster.
Successfully completed step 1/2, generating recovery snapshots for all AFM DR acting primary
filesets.
Performing step 2/2, creation of recovery output file for failback to new primary.
Successfully completed step 2/2, creation of recovery output file for failback to new primary.

File to be used with new primary cluster in next step of failback to new primary cluster:
/root//DR_Config
```

2. Transfer the newly created DR configuration file to the new primary cluster.

   ```
   scp /root//DR_Config zippleback-vm1:/root/
   ```

   The system displays output similar to the following:

   ```
   root@zippleback-vm1's password:
   DR_Config 100% 1996 2.0KB/s 00:00
   ```

3. On the new primary cluster, use the following command to create the independent filesets that will receive the data transferred from the recovery snapshots.

   ```
   mmcesdr primary failback --prep-outband-transfer --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Creating independent filesets to be used as recipients of AFM DR outband transfer of data.
   Successfully completed creating independent filesets to be used as recipients of AFM DR outband
   transfer of data.
   ```

Transfer data from recovery snapshots through outbound trucking to the newly created independent filesets before proceeding to the next step.

4. Transfer data from within the recovery snapshots of the secondary cluster to the new primary cluster.

   Note: Only one transfer is shown in the example below.

   ```
   rsync -av /gpfs/fs0/smb2/.snapshots/psnap0-newprimary-base-rpo-090B66F65623DEBF-6/*
   zippleback-vm1:/gpfs/fs0/smb2/
   ```

   The system displays output similar to the following:

   ```
   root@zippleback-vm1's password:
   sending incremental file list
   test

   sent 68 bytes received 31 bytes 15.23 bytes/sec
   total size is 0 speedup is 0.00
   ```

   Attention: When transferring files that need to also transfer GPFS extended attributes, extra steps are required. This example uses standard rsync which does not transfer extended attributes.

5. On the new primary cluster, use the following command to convert the independent filesets to primary filesets and generate a new DR configuration file that will be used on the primary cluster for the next steps and then transferred to the secondary cluster to be used in a later step.

```
mmcesdr primary failback --convert-new --output-file-path /root/ --input-file-path /root/
```

The system displays output similar to the following:

```
Performing step 1/2, conversion of independent filesets into new primary filesets to be used for AFM DR.
Successfully completed step 1/2, failback to primary on all AFM DR protected filesets.
Performing step 2/2, creation of output file for remaining failback to new primary steps.
Successfully completed step 2/2, creation of output file for remaining failback to new primary steps.

File to be used with new primary cluster in next step of failback to new primary cluster: /root//DR_Config
```

6. On the new primary cluster, use the following command.

   ```
   mmcesdr primary failback --start --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing failback to primary on all AFM DR protected filesets.
   Successfully completed failback to primary on all AFM DR protected filesets.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.
7. On the new primary cluster, use the following command one or more times until the amount of time it takes to complete the operation is less than the RPO value that you have set.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 45 Mins. 10 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```
8. On the secondary cluster (acting primary), quiesce all client operations.
9. On the new primary cluster, use the following command one more time.

   ```
   mmcesdr primary failback --apply-updates --input-file-path "/root/"
   ```

   The system displays output similar to the following:

   ```
   Performing apply updates on all AFM DR protected filesets.
   Longest elapsed time is for fileset fs1:obj_sofpolicy1 and is 0 Hrs. 0 Mins. 16 Secs.
   Successfully completed failback update on all AFM DR protected filesets.
   Depending on user load on the acting primary, this step may need to be performed again before
   stopping failback.
   ```

   Note: The **--input-file-path** parameter is optional but it might be needed if access to the configuration file is not available in the configuration fileset.
10. On the new primary cluster, use the following command to stop the failback process and convert the new primary filesets to read/write.

    ```
    mmcesdr primary failback --stop --input-file-path "/root/"
    ```

    The system displays output similar to the following:

    ```
    Performing stop of failback to primary on all AFM DR protected filesets.
    Successfully completed stop failback to primary on all AFM DR protected filesets.
    ```
11. On the new primary cluster, use the following command to restore the protocol and export services configuration information.

    ```
    mmcesdr primary restore --new-primary --file-config --restore
    ```

    Note: The `--new-primary` option must be used to ensure protocol configuration is restored correctly.

    The system displays output similar to the following:

    ```
    Restoring cluster and enabled protocol configurations/exports.
    Successfully completed restoring cluster and enabled protocol configurations/exports.

    ================================================================================
    =  If all steps completed successfully, remove and then re-create file
    ```

```
=  authentication on the Primary cluster.
=  Once this is complete, Protocol Cluster Configuration Restore will be complete.
==============================================================================
```

12. On the primary cluster, remove the file authentication and then add it again.
13. Transfer the updated DR configuration file from the new primary cluster to the secondary cluster.

    ```
    scp /root//DR_Config windwalker-vm1:/root/
    ```

    The system displays output similar to the following:

    ```
    root@windwalker-vm1's password:
    DR_Config 100% 2566 2.5KB/s 00:00
    ```

14. On the secondary cluster, use the following command to register the new primary AFM IDs to the independent filesets on the secondary cluster acting as part of the AFM DR pairs.

    ```
    mmcesdr secondary failback --post-failback-complete --new-primary --input-file-path "/root"--file-config --restore
    ```

    The system displays output similar to the following:

    ```
    Performing step 1/2, converting protected filesets back into AFM DR secondary filesets.
    Successfully completed step 1/2, converting protected filesets back into AFM DR secondary filesets.
    Performing step 2/2, restoring AFM DR-based NFS share configuration.
    Successfully completed step 2/2, restoring AFM DR-based NFS share configuration.

    ================================================================================
    = If all steps completed successfully, remove and then re-create file
    = authentication on the Secondary cluster.
    = Once this is complete, Protocol Cluster Failback will be complete.
    ================================================================================
    ```

15. On the secondary cluster, remove the file authentication and then add it again.

Backing up and restoring protocols and CES configuration information

The backup and restore capabilities of the **mmcesdr** command can be used when there is no secondary cluster and the user still wants to protect protocol and export services configuration information. The independent fileset used to store protocol and export services configuration information can be backed up using IBM Spectrum Protect software and then restored if it ever needs to be restored to the cluster.

Note: The following steps only describe how to back up and restore the protocols and CES configuration information. The actual data contained in protocol exports would need to be backed up and restored separately.

1. On the primary cluster, use the following command to back up the configuration information.

   ```
   mmcesdr primary backup
   ```

   The system displays output similar to the following:

   ```
   Performing step 1/2, configuration fileset creation/verification.
   Successfully completed step 1/2, configuration fileset creation/verification.
   Performing step 2/2, protocol and export services configuration backup.
   Successfully completed step 2/2, protocol and export services configuration backup.
   ```

   For backup, you can use IBM Spectrum Protect (formerly known as Tivoli Storage Manager) or some other tool. For example, you can use **mmbackup** as follows:

   ```
   mmbackup configuration_fileset_link_point --scope inodepace -t full
   ```

2. On the primary cluster, restore data from the off cluster storage into the configuration fileset. If **mmbackup** was used to back up the configuration fileset, the IBM Spectrum Protect command to restore is similar to the following.

   ```
   dsmc restore -subdir=yes "configuration_fileset_link_point /*"
   ```

3. On the primary cluster, use the following command to restore the configuration information.

   ```
   mmcesdr primary restore
   ```

   The system displays output similar to the following:

   ```
   Restoring cluster and enabled protocol configurations/exports.
   Successfully completed restoring cluster and enabled protocol configurations/exports.
   ```

In some cases, running the **mmcesdr primary restore** command might display the following error message:
`Saved configuration file does not exist.`. In this case, do the following:

- If this cluster is part of a Protocols DR relationship, place a copy of the DR configuration file at a specified location and run the **mmcesdr primary restore** command again using the `--input-file-path` option.
- If this cluster is not part of a Protocols DR relationship, run this command again with the `--file-config --restore` option to force restoring the file configuration information. The system displays output similar to the following:

```
# mmcesdr primary restore --file-config --restore
Restoring cluster and enabled protocol configurations/exports.
Successfully completed restoring cluster and enabled protocol configurations/exports.

================================================================================
=  If all steps completed successfully, remove and then re-create file
=  authentication on the Primary cluster.
=  Once this is complete, Protocol Cluster Configuration Restore will be complete.
================================================================================
```

4. On the primary cluster, remove the file authentication and then add it again.

   Note: If you want to perform a restore as part of a failback (either to an old primary cluster or a new primary cluster) and want to re-create the file configuration/exports, use one of the following commands:
   mmcesdr primary restore
   or
   mmcesdr primary restore --file-config --recreate.

Updating protocols and CES configuration information

You can use the following command to update the backed up configuration information for object, NFS, and SMB protocols, and CES.

```
mmcesdr primary update {--obj | --nfs | --smb | --ces}
```

Note: No output is generated by the command, because this command is designed to be scripted and run on a regular basis or run as a part of a callback. The update command can only be used to update the primary configuration after Protocols DR has been configured. If no secondary cluster exists, use the **mmcesdr primary backup** command to back up configuration for later restore.

Protocols and cluster configuration data required for disaster recovery

For protocols cluster disaster recovery, data needs to be collected for failover, failback, backup, or restore from the respective protocol, for authentication, and for cluster wide information.

Use the following information to collect the data required for protocols cluster disaster recovery.

Object data required for protocols cluster DR

Data required for object protocol in case of a disaster recovery scenario is as follows.

Note: IBM Spectrum Scale 4.2 and later versions for object storage supports either AFM DR-based protection or multi-region object deployment , but not both. If multi-region object deployment is enabled, no object data or configuration information is protected through protocols cluster DR.

In addition to the standard object fileset, independent filesets are created for each object policy that is created. This in turn creates additional CCR files that are listed here. All of these additional filesets and additional configuration information are protected, if IBM Spectrum Scale for object storage is using the AFM DR-based protection and not multi-region object deployment.

You can determine all object filesets using the **`mmobj policy list`** command.

The object related files in CCR that need to be backed up are as follows:

1. `account.builder`
2. `account.ring.gz`
3. `account-server.conf`
4. `container.builder`
5. `container-reconciler.conf`
6. `container.ring.gz`
7. `container-server.conf`
8. `keystone_ssl.tar`
9. `keystone.tar`
10. `object.builder`
11. `object<`*`index`*`>.builder` for each storage policy, where <*index*> is a unique number representing the storage policy
12. `object-expirer.conf`
13. `object.ring.gz`
14. `object<`*`index`*`>.ring.gz` for each storage policy, where <*index*> is a unique number representing the storage policy
15. `object-server.conf`
16. `object-server-sof.conf`
17. `openrc`
18. `objRingVersion`
19. `postgresql-obj.service`
20. `proxy-server.conf`
21. `spectrum-scale-object.conf`
22. `spectrum-scale-object-policies.conf`
23. `spectrum-scale-objectizer.conf`
24. `spectrum-scale-local-region.conf`
25. `spectrum-scale-global-region.conf`
26. `spectrum-scale-compression-scheduler.conf`
27. `spectrum-scale-global-region.conf`
28. `spectrum-scale-compression-status.stat`
29. `swift.conf`
30. `swift.tar`

The following CCR files also need to be backed up FOR local object authentication :

- keystone.conf
- keystone-paste.ini
- logging.conf

For a list of object authentication related CCR files and variables that need to be backed up, see "Authentication related data required for protocols cluster DR" on page 286.

You can back up Postgres database files as follows:

1. Take the file system snapshot of the shared root file system.
2. Create a tar or a zip file of the `object` directory within the CES shared root file system and everything underneath this directory.
3. Save the tar or the zip file
4. Delete the snapshot.

Failover steps for object configuration if you are using local authentication for object

Use the following steps on a protocol node in the secondary cluster to fail over the object configuration data. Use this set of steps if you are using local authentication for object.

You need to determine the following two values in the secondary cluster:

- Cluster host name: This is the DNS value which returns a CES IP address from the pool of addresses in the secondary cluster.
- Object database node: This is the CES IP address which is configured to run the postgresql-obj database for object services. You can find this value as the address designated as the `object_database_node` in the output of the **`mmces address list`** command. For example:

```
mmces address list

Node  Daemon node name     IP address     CES IP address list
---------------------------------------------------------------
  1   vm1.lab.gpfs.net     9.1.2.1        192.168.1.1
  2   vm2.lab.gpfs.net     9.1.2.2        192.168.1.2
  3   vm3.lab.gpfs.net     9.1.2.3        192.168.1.3(object_database_node)
```

Important:

The following object steps must be run on the node designated as `object_database_node` in the secondary cluster. This ensures that postgresql-obj and Keystone servers can connect during this configuration process.

1. Stop the object protocol services using the following command:
   ```
   mmces service stop OBJ --all
   ```
2. Make two changes in the preserved Cluster Configuration Repository (CCR) configuration to update it for the DR environment:
   a. The `keystone.conf` file: Edit this file to change the database connection address to `object_database_node` of the secondary cluster. For example:
   ```
   Change this:
   [database]
   connection = postgresql://keystone:password@192.168.56.1/keystone

   to this:
   [database]
   connection = postgresql://keystone:password@192.168.1.3/keystone
   ```
   Note: If the **`mmcesdr`** command is used to save the protocol cluster configuration, then the preserved copy of the `keystone.conf` file is located at the following location:

   `CES_shared_root_mount_point``/.async_dr/Failover_Config/Object_Config/latest/ccr_files/keystone.conf`

   You can edit the file directly to make this change or use the **`openstack-config`** command. For example, first retrieve the current value using `get`, and then update it using the `set` option:
   ```
   openstack-config --get keystone.conf  database connection \
   postgresql://keystone:passw0rd@192.168.56.1/keystone
   ```

```
openstack-config --set keystone.conf  database connection \
postgresql://keystone:passw0rd@192.168.1.3/keystone

openstack-config --get keystone.conf  database connection \
postgresql://keystone:passw0rd@192.168.1.3/keystone
```

b. The *ks_dns_name* variable: This is one of the object related variables that was originally preserved from CCR. Modify the value of this variable to the cluster hostname of the secondary cluster.

   Note: If the **mmcesdr** command is used to save the protocol cluster configuration, then the preserved copy of the *ks_dns_name* variable is located as a line in the following file:

   CES_shared_root_mount_point/.async_dr/Failover_Config/Object_Config/latest/ccr_vars/file_for_ccr_variables.txt

   Change the value of the variable in this preserved copy of the file.

3. [Optional] If spectrum-scale-localRegion.conf exists from CCR, change the cluster hostname and cluster_id properties to the cluster host name and cluster id as shown in the output of the **mmlscluster** command.
4. Restore the Postgres database information to the shared root directory. The directory needs to be first cleaned out before the archive is restored. This can be done with commands similar to the following, assuming that the directory was tar/zip when it was backed up:
   a. Delete the old Postgres data:
   ```
   rm -rf <shared_root_location>/object/keystone/*
   ```
   b. Verify that the shared root directory is empty:
   ```
   ls <shared_root_location>
   ```
   c. Restore the current Postgres database:
   ```
   tar xzf <tar_file_name>.gz -C <shared_root_location>
   ```
   d. Delete the process status file from the primary:
   ```
   rm -rf <shared_root_location>/object/keystone/postmaster.pid
   ```
   e. List the Postgres files:
   ```
   ls <shared_root_location>
   ```
5. Restore all object configuration CCR files, including **keystone.conf** with the modification for object_database_node, with a command similar to the following:
   ```
   mmccr fput <file> <location>/<file>
   ```
6. If object policies are present, restore all of the object policy related CCR files.
7. Restore all object configuration CCR variables, including *ks_dns_name* with the modification for cluster host name, with a command similar to the following:
   ```
   mmccr vput name value
   ```
8. Start the Postgres database and verify that it is running successfully using commands similar to the following:
   ```
   systemctl start postgresql-obj
   sleep 5
   systemctl status postgresql-obj
   ```
   ```
   These commands generate output similar to:
   postgresql-obj.service - postgresql-obj database server
   Loaded: loaded (/etc/systemd/system/postgresql-obj.service; disabled)
   Active: active (running) since Thu 2015-05-28 19:00:17 EDT; 20h ago
   ```
9. Load openrc definitions by running the following command.
   ```
   source /root/openrc
   ```
10. Run the following command for the value of the api_v3 pipeline. Save the output of this command into the variable <savedAPI_V3Pipeline>.

`mmobj config list --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline` . If the value of the api_v3 pipeline does not contain the string **admin_token_auth**, then do the following:

a. Make a note that the api_v3 pipeline had to be updated.

b. Generate the new value of the api_v3 pipeline by inserting the string **admin_token_auth** directly after the string **token_auth** in the saved copy of the api_v3 pipeline.

c. Change the api_v3 pipeline value using the command: `mmobj config change --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline --value <newAPI_V3Pipeline>`, where *<newAPI_V3Pipeline>* is the updated value from the previous step.

d. List the updated api_v3 pipeline value by running the following command and ensure the **admin_token_auth** string is present: `mmobj config list --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline`.

11. Save the restored CCR copy of the `keystone.conf` to the local location using a command similar to **`mmccr fget keystone.conf /etc/keystone/keystone.conf`**. Also, update the owner and the group of this file using the following command: **`chown keystone:keystone /etc/keystone/keystone.conf`**.

    Note: If SSL is enabled, SSL certificates must be in place when you are saving `keystone.conf` from another cluster.

12. If the **`DEFAULT admin_token`** is set, save its current value by using a command similar to the following:

```
openstack-config --get /etc/keystone/keystone.conf DEFAULT admin_token
```

    If a value is returned from the above command, save it because it will need to be restored later.

13. In the `keystone.conf` file, set **`admin_token`** to `ADMIN` using the **`openstack-config`** command as follows.

```
openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_token ADMIN
```

14. Set the following environment variables.

```
export OS_TOKEN=ADMIN     # The value from admin_token
export OS_URL="http://127.0.0.1:35357/v3"
```

15. Start Keystone services and get the list of endpoint definitions using commands similar to the following.

```
systemctl start httpd
sleep 5
openstack endpoint list
```

```
These commands generate output similar to:
+------------+--------+--------------+--------------+---------+-----------+-----------------------------------------------|
| ID         | Region | Service Name | Service Type | Enabled | Interface | URL                                           |
+------------+--------+--------------+--------------+---------+-----------+-----------------------------------------------|
| c36e..9da5 | None   | keystone     | identity     | True    | public    | http://specscaleswift:5000/                   |
| f4d6..b040 | None   | keystone     | identity     | True    | internal  | http://specscaleswift:35357/                  |
| d390..0bf6 | None   | keystone     | identity     | True    | admin     | http://specscaleswift:35357/                  |
| 2e63..f023 | None   | swift        | object-store | True    | public    | http://specscaleswift:8080/v1/AUTH_%(tenant_id)s |
| cd37..9597 | None   | swift        | object-store | True    | internal  | http://specscaleswift:8080/v1/AUTH_%(tenant_id)s |
| a349..58ef | None   | swift        | object-store | True    | admin     | http://specscaleswift:8080                    |
+------------+--------+--------------+--------------+---------+-----------+-----------------------------------------------|
```

16. Update the host name specified in the endpoint definitions in the URL value from the endpoint list. The values in the endpoint table might have the cluster host name (`ces1` in this example) from the primary system. They need to be updated for the cluster host name in the DR environment. In some environments, the cluster host name is the same between the primary and secondary clusters. If that is the case, skip this step.

    a. Delete the existing endpoints with the incorrect cluster host name. For each of the endpoints, use the ID value to delete the endpoint. For example, use a command similar to this to delete each of the six endpoints:

```
openstack endpoint delete e149
```

    b. Recreate the endpoints with the cluster host name of the secondary cluster using the following commands:

    The *CHN* variable in the following commands is the cluster host name for the secondary cluster.

```
openstack endpoint create identity public "http://$CHN:5000/v3"
openstack endpoint create identity internal "http://$CHN:35357/v3"
openstack endpoint create identity admin "http://$CHN:35357/v3"
openstack endpoint create object-store public "http://$CHN:8080/v1/AUTH_%(tenant_id)s"
openstack endpoint create object-store internal "http://$CHN:8080/v1/AUTH_%(tenant_id)s"
openstack endpoint create object-store admin "http://$CHN:8080"
```

c. Verify that the endpoints are now using the correct cluster host name using the following command:

```
openstack endpoint list
```

17. If the api_v3 pipeline had to be updated previously, return it to its original value by running the following command: `mmobj config change --ccrfile keystone-paste.ini --section pipeline:api_v3 --property pipeline --value <savedAPI_V3Pipeline>` , where *<savedAPI_V3Pipeline>* is the value of the api_v3 pipeline saved above.
18. Depending on whether a value for the DEFAULT admin_token was previously set, do one of the following:
   - If a value for the **DEFAULT admin_token** was previously set, reset it to that value using a command similar to the following:

     ```
     openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_token ${currentAdminToken}
     ```
   - If there was no previous value for the **DEFAULT admin_token**, delete the value that was set to convert the endpoint definitions using a command similar to the following:

     ```
     openstack-config --del /etc/keystone/keystone.conf DEFAULT admin_token
     ```
19. Stop the services that were manually started earlier and clean up using the following commands:

   ```
   systemctl stop httpd
   systemctl stop postgresql-obj
   unset OS_URL
   unset OS_TOKEN
   ```
20. Start the object protocol services using the following command:

   ```
   mmces service start OBJ --all
   ```

Failover steps for object configuration if you are not using local authentication for object:

Use the following steps on a protocol node on a secondary cluster to fail over object configuration data. Use this set of steps if you are not using local authentication for object.

1. Stop the object protocol services using the following command:

   ```
   mmces service stop OBJ --all
   ```
2. Restore the Postgres database information to the shared root directory. The directory needs to be first cleaned out before the archive is restored. This can be done with commands similar to the following, assuming that the directory was tar/zip when it was backed up:
   a. Delete the old Postgres data:

      ```
      rm -rf <shared_root_location>/object/keystone/*
      ```
   b. Verify that the shared root directory is empty:

      ```
      ls <shared_root_location>
      ```
   c. Restore the current Postgres database:

      ```
      tar xzf <tar_file_name>.gz -C <shared_root_location>
      ```
   d. Delete the process status file from the primary:

      ```
      rm -rf <shared_root_location>/object/keystone/postmaster.pid
      ```
   e. List the Postgres files:

      ```
      ls <shared_root_location>
      ```
3. Restore all object configuration CCR files with a command similar to the following:

   ```
   mmccr fput <file> <location>/<file>
   ```
4. If object policies are present, restore all of the object policy related CCR files.

5. Restore all object configuration CCR variables with a command similar to the following:
   ```
   mmccr vput name value
   ```
6. Start the object protocol services using the following command:
   ```
   mmces service start OBJ --all
   ```

Failback or restore steps for object configuration

Use the following steps on a protocol node in the primary cluster to fail back or restore the object configuration data.

You need to determine the `object_database_node` on the primary cluster once repaired or replaced.

- Object database node: This is the CES IP address which is configured to run the postgresql-obj database for object services. You can find this value as the address designated as the `object_database_node` in the output of the **`mmces address list`** command. For example:
  ```
  mmces address list

  Node  Daemon node name     IP address     CES IP address list
  ---------------------------------------------------------------
     1  vm1.lab.gpfs.net     9.1.2.1        192.168.1.1
     2  vm2.lab.gpfs.net     9.1.2.2        192.168.1.2
     3  vm3.lab.gpfs.net     9.1.2.3        192.168.1.3(object_database_node)
  ```

Important:

The following object steps must be run on the node designated as `object_database_node` in the primary cluster. This ensures that postgresql-obj and Keystone servers can connect during this configuration process.

1. Stop the object protocol services using the following command:
   ```
   mmces service stop OBJ --all
   ```
2. Restore the Postgres database information to the shared root directory. The directory needs to be first cleaned out before the archive is restored. This can be done with commands similar to the following, assuming that the directory was tar/zip when it was backed up:
   a. Delete the old Postgres data:
      ```
      rm -rf <shared_root_location>/object/keystone/*
      ```
   b. Verify that the shared root directory is empty:
      ```
      ls <shared_root_location>
      ```
   c. Restore the current Postgres database:
      ```
      tar xzf <tar_file_name>.gz -C <shared_root_location>
      ```
   d. Delete the process status file from the primary:
      ```
      rm -rf <shared_root_location>/object/keystone/postmaster.pid
      ```
   e. List the Postgres files:
      ```
      ls <shared_root_location>
      ```
3. Restore all object configuration CCR files with a command similar to the following:
   ```
   mmccr fput <file> <location>/<file>
   ```
4. If object policies are present, restore all of the object policy related CCR files.
5. Restore all object configuration CCR variables with a command similar to the following:
   ```
   mmccr vput name value
   ```
6. Start the object protocol services using the following command:
   ```
   mmces service start OBJ --all
   ```

SMB data required for protocols cluster DR

Data required for the SMB protocol in case of a disaster recovery scenario is as follows.

You can determine the SMB exports using the **`mmsmb export list`** command.

The SMB protocol related files that need to be backed up are as follows.

- `account_policy.tdb`
- `autorid.tdb`[1]
- `group_mapping.tdb`
- `passdb.tdb`
- `registry.tdb`
- `secrets.tdb`
- `share_info.tdb`
- `ctdb.tdb`

Note: [1] This file is required only if the file authentication is configured with Active Directory.

The following information is common for all of these files:

- The type of these files is persistent TDB.
- They are not in the cluster configuration repository (CCR).
- Their location is `/var/lib/ctdb/persistent` on all protocol nodes.
- Their contents are same on all the nodes.
- Their name consists of TDB name + node specific extension. For example: `registry.tdb.0`

The private Kerberos configuration files available at the following location also need to be backed up: `/var/lib/samba/smb_krb5/`. You can copy these files from this location and save them.

Failover steps for the SMB protocol

Use the following steps on a protocol node in the secondary cluster to fail over the SMB protocol configuration.

1. Before stopping the SMB services, make a note of the node number that will be used to restore TDB files because the node numbers are not available when SMB services are stopped.
2. Stop the SMB services using the following command:
   ```
   mmces service stop SMB --all
   ```
3. Issue the following command to stop the NFS service:
   ```
   mmces service stop NFS --all
   ```
4. Remove the contents of the `/var/lib/ctdb/persistent` directory on all protocol nodes.
5. Restore the previously saved TDB files to one of the protocol nodes and place them in the `/var/lib/ctdb/persistent` directory for the node where the node number was saved.

   However, when copying the files to that directory on the node, replace the `X.bak`, where X represents the node number where the files were copied from, with the new node number. It is crucial that each of these files ends with `.tdb.Y`, where Y is the node number that was saved and the node number where the files are being restored. These files only need to be put into one of the nodes and when the SMB processes are started again they are copied around to the other nodes properly.
6. Remove the contents of the `/var/lib/samba/smb_krb5` directory on all the protocol nodes.
7. Restore the saved contents of `smb_krb5` to the `/var/lib/samba/smb_krb5/` directory on one of the protocol nodes. No special extension needs to be altered in this case.
8. Start the SMB services using the following command:
   ```
   mmces service start SMB --all
   ```

9. Issue the following command to start the NFS service:
   ```
   mmces service start NFS --all
   ```
10. Remove SMB exports that are not protected using AFM DR independent filesets.

Failback or restore steps for the SMB protocol

Use the following steps on a protocol node in the primary cluster, once repaired or replaced, to fail back or restore the SMB protocol configuration.

1. Stop the NFS services using the following command:
   ```
   mmces service stop NFS --all
   ```
2. Stop the SMB services using the following command:
   ```
   mmces service stop smb --all
   ```
3. Delete all files from the `/var/lib/ctdb/persistent` directory on all protocol nodes.
4. Restore all required persistent TDB files from the saved configuration location to the `/var/lib/ctdb/persistent` directory on one of the protocol nodes. Ensure that you append the node number to the end of file names.
5. Delete all private Kerberos configuration files in the `/var/lib/samba/smb_krb5/` directory on all protocol nodes.
6. Restore private Kerberos configuration files to the `/var/lib/samba/smb_krb5/` directory on one of the protocol nodes.
7. On the failback cluster, start the SMB services using the following command:
   ```
   mmces service start smb --all
   ```
8. Issue the following command on the failback cluster to start the NFS service:
   ```
   mmces service start NFS --all
   ```

NFS data required for protocols cluster DR

Data required for NFS protocol in case of a disaster recovery scenario is as follows.

You can find the NFS exports using the following command:
```
mmnfs export list
```

If the NFS exports are independent filesets, AFM based Disaster Recovery (AFM DR) can be used to replicate the data.

The NFS protocol related CCR files that need to be backed up are as follows.
- `gpfs.ganesha.main.conf`
- `gpfs.ganesha.nfsd.conf`
- `gpfs.ganesha.log.conf`
- `gpfs.ganesha.exports.conf`
- `gpfs.ganesha.statdargs.conf`

The following NFS protocol related CCR variable needs to be backed up.
- *nextexportid*

Failover steps for the NFS protocol

Use the following steps on one of the protocol nodes in the secondary cluster to fail over the NFS protocol configuration.

1. Stop the NFS services using the following command:
   ```
   mmces service stop NFS --all
   ```
2. Edit the saved `gpfs.ganesha.exports.conf` export configuration file to remove all exports that are not protected through AFM DR independent filesets.

3. Restore the NFS related CCR files. For a list of these files, see "NFS data required for protocols cluster DR" on page 285
4. Restore the *nextexportid* CCR variable.
5. Load the exports file using the following command:
   ```
   mmnfs export load /<Path_to_CCR_files>/gpfs.ganesha.exports.conf
   ```
6. Start the NFS services using the following command:
   ```
   mmces service start NFS --all
   ```

Failback or restore steps for the NFS protocol

Use the following steps on a protocol node in the primary cluster, once repaired or replaced, to fail back or restore the NFS protocol configuration.

1. Stop the NFS services using the following command:
   ```
   mmces service stop NFS --all
   ```
2. Restore the NFS related CCR files. For a list of these files, see "NFS data required for protocols cluster DR" on page 285.
3. Restore the *nextexportid* CCR variable.
4. Load the exports file using the following command:
   ```
   mmnfs export load /<Path_to_saved_CCR_files>/gpfs.ganesha.exports.conf
   ```
5. Start the NFS services using the following command:
   ```
   mmces service start NFS --all
   ```

Authentication related data required for protocols cluster DR

Authentication data required in case of a disaster recovery scenario is as follows.

The following authentication related CCR file needs to be backed up for disaster recovery.
- `authccr`

File authentication related data

The following CCR variable needs to be backed up for file authentication:
- *FILE_AUTH_TYPE*

Depending on the file authentication scheme you are using, additional files need to be backed up.

LDAP for file authentication:
- `SSSD_CONF`
- `LDAP_CONF`
- `KRB5_CONF` [1]
- `KRB5_KEYTAB` [1]
- `LDAP_TLS_CACERT` [1]

Active Directory (AD) for file authentication:
- `KRB5_CONF`
- `KRB5_KEYTAB` [1]

NIS for file authentication:
- `SSSD_CONF`
- `YP_CONF`

Note: [1] This file is not always present.

Object authentication related data

The object authentication related files in CCR that need to be backed up are as follows:
- `keystone.conf`
- `keystone-paste.ini`
- `logging.conf`
- `wsgi-keystone.conf`

The object authentication related variables in CCR that need to be backed up are as follows:
- *OBJECT_AUTH_TYPE*
- *PREV_OBJECT_AUTH_TYPE*

  This variable may not be present if the authentication type has not changed.
- *OBJECT_IDMAPDELETE*
- *ks_db_type*
- *ks_db_user*
- *ks_db_user_pwd*
- *ks_dns_name*

Failover steps for authentication data

Use the following steps on a protocol node in the secondary cluster to fail over the authentication configuration.

Note: The object authentication does not need to be removed and re-added as part of failover, failback, or restore.
1. Save the current file authentication information on the secondary cluster.
2. Remove file authentication from the secondary cluster.
3. Restore file authentication on the secondary cluster based on the information saved in step 1.

Failback steps for authentication data

Use the following steps on a protocol node in the primary cluster, once repaired or replaced, to fail back the authentication configuration.

Note: The object authentication does not need to be removed and re-added as part of failover, failback, or restore.
1. Save the current file authentication information on the primary cluster.
2. Remove file authentication from the primary cluster.
3. Restore file authentication on the primary cluster based on the information saved in step 1.

Restore steps for authentication data:

Use the following steps on a protocol node in the primary cluster and the secondary cluster to restore the authentication configuration.

Note: The object authentication does not need to be removed and re-added as part of failover, failback, or restore.
1. Save the current file authentication information on the primary cluster.
2. Remove file authentication from the primary cluster.
3. Restore file authentication on the primary cluster based on the information saved in step 1.
4. Save the current file authentication information on the secondary cluster.
5. Remove file authentication from the secondary cluster.

6. Restore file authentication on the secondary cluster based on the information saved in step 4.

CES data required for protocols cluster DR

Cluster Export Services (CES) data required in case of a disaster recovery scenario is as follows.

Cluster Configuration Repository (CCR) files that need to be backed up for CES in a disaster recovery scenario are as follows:

- `mmsdrfs`
- `cesiplist`

Failover steps for CES

No Cluster Export Services (CES) configuration information is restored on fail over. This is because this information is typically cluster specific and it would interfere with the proper operating of the secondary cluster.

Failback or recovery steps for CES

Use the following steps on a protocol node in the primary cluster to fail back or recover the CES configuration.

1. Restore the `cesiplist` file.
2. For each protocol node listed in the stored, backup copy of the `mmsdrfs` file, verify that the node on the primary cluster is also configured as a protocol node. If not, use the **mmchnode** to enable the node as a protocol node.
3. For each of the following CES parameters in the stored, backup copy of the `mmsdrfs` file, verify that the value is the same on the primary cluster. If not, use the **mmchconfig** to update the configuration value.
   - **cesSharedRoot**
   - **cesAddressPool**
   - **cesServices**
   - **cifsBypassTraversalChecking**
   - **syncSambaMetadataOps**
   - **cifsBypassShareLocksOnRename**

Chapter 17. File Placement Optimizer

GPFS File Placement Optimizer (FPO) is a set of features that allow GPFS to operate efficiently in a system based on a shared nothing architecture. It is useful for big data applications that process massive amounts of data.

Note: This feature is available with IBM Spectrum Scale Express Edition, Standard Edition or higher.

FPO uses the following entities and policies:

Chunks

A chunk is a logical grouping of blocks that allows the grouping to behave like one large block, useful for applications that need high sequential bandwidth. Chunks are specified by a block group factor that dictates how many file system blocks are laid out sequentially on disk to behave like a large block. Different Chunk size can be defined by block group factor on file level or defined globally on a storage pool by default.

On the file level, the block group factor can be specified by the **--block-group-factor** argument of the **mmchattr** command. You can also specify the block group factor by the **setBGF** argument of the **mmchpolicy** and **mmapplypolicy** command.The range of the block group factor is 1 - 1024. The default value is 1. You can also specify the block group factor through the **blockGroupFactor** argument in a storage pool stanza (as input to the **mmadddisk** or **mmcrfs** command).

The effective chunk size is a multiplication of Block Group Factor and GPFS block size. For example, setting block size to 1 MB and block group factor to 128 leads to an effective large block size of 128 MB.

See the following command descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:

- **mmadddisk**
- **mmchattr**
- **mmcrfs**

Extended failure groups

A failure group is defined as a set of disks that share a common point of failure that might cause them all to become simultaneously unavailable. Traditionally, GPFS failure groups are identified by simple integers. In an FPO-enabled environment, a failure group might be specified as not just a single number, but as a vector of up to three comma-separated numbers. This vector conveys topology information that GPFS exploits when making data placement decisions.

In general, a topology vector is a way for the user to specify which disks are closer together and which are farther away. In practice, the three elements of the failure group topology vector might represent the rack number of a disk, a position within the rack, and a node number. For example, the topology vector 2,1,0 identifies rack 2, bottom half, first node.

The Data block placement decisions about the disk selection for data replica are made by GPFS based on the Failure group. When considering two disks for striping or replica placement purposes, it is important to understand the following:

- Disks that differ in the first of the three numbers are farthest apart (as they are in different racks).
- Disks that have the same first number but differ in the second number are closer (as they are in the same rack, but in different halves).
- Disks that differ only in the third number reside in different nodes in the same half of the same rack.
- Only disks that have all three numbers in common reside in the same node.

The data block placement decisions are also affected by the level of replication and the value of the **writeAffinityDepth** parameter. For example, when using replication 3, GPFS might place two replicas far apart (different racks) to minimize chances of losing both. However, the third replica can be placed close to one of the others (same rack, but different half), to reduce network traffic between racks when writing the three replicas.

To specify the topology vector that identifies a failure group, you use the **failureGroup=***FailureGroup* attribute in an NSD stanza (as input to the **mmadddisk** or **mmcrfs** command).

See the following command descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:

- **mmadddisk**
- **mmcrfs**

Write affinity depth

Write affinity depth is a policy that allows the application to determine the layout of a file in the cluster to optimize for typical access patterns. The write affinity is specified by a depth that indicates the number of localized copies (as opposed to wide striped). It can be specified at the storage pool or file level. The enabling of Write affinity depth, indicates that the first replica is being written on the node where the writing is triggered. It also indicates, the second and third replica (if any) are being written on the other node disks.

To specify write affinity depth, you use the **writeAffinityDepth** attribute in a storage pool stanza (as input to the **mmadddisk** or **mmcrfs** command) or the **--write-affinity-depth** argument of the **mmchattr** command. You can also use **--block-group-factor** argument of the **mmchpool** command to change a storage pool's block group factor

A write affinity depth of 0 indicates that each replica is to be striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. By default, the unit of striping is a block; however, if the block group factor is specified in order to exploit chunks, the unit of striping is a chunk.

A write affinity depth of 1 indicates that the first copy is written to the writer node. The second copy is written to a different rack. The third copy is written to the same rack as the second copy, but on a different half (which can be composed of several nodes).

A write affinity depth of 2 indicates that the first copy is written to the writer node. The second copy is written to the same rack as the first copy, but on a different half (which can be composed of several nodes). The target node is determined by a hash value on the fileset ID of the file, or it is chosen randomly if the file does not belong to any fileset. The third copy is striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group.

This behavior can be altered on an individual file basis by using the **--write-affinity-failure-group** option of the **mmchattr** command.

Note: In fileset level, Write affinity depth of 2 is design to assign (write) all the files in a fileset to the same second-replica node. However, this behavior depends on node status in the cluster. After a node is added to or deleted from a cluster, a different node might be selected as the second replica for files in a fileset.

See the description of storage pool stanzas that follows. Also, see the following command descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:

- **mmadddisk**
- **mmchattr**
- **mmcrfs**
- **mmchpolicy**
- **mmapplypolicy**

- **mmchpool**

Write affinity failure group

Write affinity failure group is a policy that indicates the range of nodes (in a shared nothing architecture) where replicas of blocks in a particular file are to be written. The policy allows the application to determine the layout of a file in the cluster to optimize for typical access patterns.

You specify the write affinity failure group through the **write-affinity-failure-group** *WafgValueString* attribute of the **mmchattr** command. You can also specify write affinity failure group through the **setWADFG** attribute of the **mmchpolicy** and **mmapplypolicy** command. Failure group topology vector ranges specify the nodes, and the specification is repeated for each replica of the blocks in a file.

For example, the attribute `1,1,1:2;2,1,1:2;2,0,3:4` indicates:

- The first replica is on rack 1, rack location 1, nodes 1 or 2.
- The second replica is on rack 2, rack location 1, nodes 1 or 2.
- The third replica is on rack 2, rack location 0, nodes 3 or 4.

The default policy is a null specification. This default policy indicates that each replica must follow the storage pool or the file-write affinity depth (WAD) definition for data placement. Not wide striped over all disks.

When data in an FPO pool is backed up in a TSM server and then restored, the original placement map is broken unless you set the write affinity failure group for each file before backup.

Note: To change the failure group in a write-affinity–enabled storage pool, you must use the **mmdeldisk** and **mmadddisk** commands; you cannot use **mmchdisk** to change it directly.

See the following command descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:

- **mmchpolicy**
- **mmapplypolicy**
- **mmchattr**

Enabling the FPO features

To efficiently support write affinity and the rest of the FPO features, GPFS internally requires the creation of special allocation map formats. When you create a storage pool that is to contain files that make use of FPO features, you must specify **allowWriteAffinity=yes** in the storage pool stanza.

To enable the policy to read from preferred replicas, issue one of the following commands:

- To specify that the policy read from the first replica, regardless of whether there is a replica on the disk, default to or issue the following:

  ```
  mmchconfig readReplicaPolicy=default
  ```
- To specify that the policy read replicas from the local disk, if the local disk has data, issue the following:

  ```
  mmchconfig readReplicaPolicy=local
  ```
- To specify that the policy read replicas from the fastest disk to read from based on the disk's read I/O statistics, issue the following:

```
mmchconfig readReplicaPolicy=fastest
```

See the description of storage pool stanzas that follows. Also, see the following command descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:

- **mmadddisk**
- **mmchconfig**
- **mmcrfs**

Storage pool stanzas

Storage pool stanzas are used to specify the type of layout map and write affinity depth, and to enable write affinity, for each storage pool.

Storage pool stanzas have the following format:

```
%pool:
  pool=StoragePoolName
  blockSize=BlockSize
  usage={dataOnly | metadataOnly | dataAndMetadata}
  layoutMap={scatter | cluster}
  allowWriteAffinity={yes | no}
  writeAffinityDepth={0 | 1 | 2}
  blockGroupFactor=BlockGroupFactor
```

See the following command descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:

- **mmadddisk**
- **mmcrfs**

Recovery from disk failure

A typical shared nothing cluster is built with nodes that have direct-attached disks. Disks are not shared between nodes as in a regular GPFS cluster, so if the node is inaccessible, its disks are also inaccessible. GPFS provides means for automatic recovery from these and similar common disk failure situations.

The following command sets up and activates the disk recovery features:

```
mmchconfig restripeOnDiskFailure=yes -i
```

Whether a file system went through a recovery is determined by the max replication values for the file system. If the **mmlsfs -M** or **-R** value is greater than one, then the recovery code is run. The recovery actions are asynchronous and GPFS continues its processing while the recovery attempts take place. The results from the recovery actions and any errors that are encountered are recorded in the GPFS logs.

Two more parameters are available for fine-tuning the recovery process:

```
mmchconfig metadataDiskWaitTimeForRecovery=seconds
mmchconfig dataDiskWaitTimeForRecovery=seconds
```

The default value for **metadataDiskWaitTimeForRecovery** is 1800 seconds. The default value for **dataDiskWaitTimeForRecovery** is 3600 seconds.

See the following command description in the *IBM Spectrum Scale: Administration and Programming Reference*:

- **mmchconfig**

Distributing data across a cluster

You can distribute data uniformly across a cluster.

Following are the possible ways to distribute the data:

- Import the data through a node that does not have any attached NSD and takes the role as a GPFS client node in the cluster. This ensures that the data is distributed evenly across all failure groups and all nodes within a failure group.
- Use a write affinity depth of 0 across the cluster.
- Make every GPFS node an ingest node and deliver data equally across all ingest nodes. However, this strategy is expensive in terms of implementation.

Ideally, all the failure groups must have an equal number of disks with roughly equal capacity. If one failure group is much smaller than the rest, it is likely to fill up faster than the others, and this complicates rebalancing actions.

After the initial ingesting of data, the cluster might be unbalanced. In such a situation, use the **mmrestripefs** command with the **-b** option to rebalance the data.

Note: For FPO users, the **mmrestripefs -b** command breaks the original data placement that follows the data locality rule.

FPO pool file placement and AFM

For AFM home or cache, an FPO pool file that is written on the local side is placed according to the write affinity depth and write affinity failure group definitions of the local side.

When a file is synced from home to cache, it follows the same FPO placement rule as when written from the gateway node in the cache cluster. When a file is synced from cache to home, it follows the same FPO data placement rule as when written from the NFS server in the home cluster.

To retain the same file placement at AFM home and cache, ensure that each has the same cluster configuration and set the write affinity failure group for each file. If the home and cache cluster have different configurations, such as the disk number, node number, or fail group, then the data locality might be broken.

Configuring FPO

Follow the steps listed in the *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* to install the GPFS RPMs and build the portability layer on all nodes in the cluster. You can configure password-less SSH for root user across all GPFS nodes. However, in cases of special security control, you can configure at least one node for the root user to access all GPFS nodes in a password-less mode. GPFS commands can be run only over these nodes.

For OS with Linux kernel 2.6, enter following commands on all GPFS nodes that are set as root to set vm.min_free_bytes :

```
# TOTAL_MEM=$(cat /proc/meminfo | grep MemTotal | tr -d \"[:alpha:]\" | tr -d \"[:punct:]\" | tr
-d \"[:blank:]\") # VM_MIN_FREE_KB=$((${TOTAL_MEM}*6/100))
```

```
# echo "vm.min_free_kbytes = $VM_MIN_FREE_KB" >> /etc/sysctl.conf # sysctl -p
```

```
# sysctl -a | grep vm.min_free_kbytes
```

Configuring IBM Spectrum Scale Clusters

All GPFS configuration steps must be performed as the root user. These steps need to be executed only on one node, not on all nodes.

Create the GPFS Cluster

The GPFS node file defines all nodes in the cluster and some of the roles.

Create the GPFS cluster with **node11** as the primary and **node21** as the secondary cluster configuration server. Set the *-A* flag to automatically start GPFS daemons when the OS is started.

```
# mmcrcluster -A -C gpfs-cluster -p node11 -s node21 -N nodefile -r $(which ssh) -R $(which scp)
```

Use the **mmlscluster** command to view the cluster.

Apply IBM Spectrum Scale license

All GPFS nodes require a license designation before they can be used. The FPO feature introduced a dedicated PFS license class **fpo**. In a GPFS FPO cluster, all quorum and manager nodes require a server license. Based on the sample environment, **node11**, **node21**, and **node31** require a server license. The other nodes require an **fpo** license.

```
# mmchlicense server --accept -N node11,node21,node31
```

```
# mmchlicense fpo --accept -N node12,node13,node14,node15,node16,node22,node23,node24,node25,node26,
node32,node33,node34,node35,node36
```

Use the **mmlslicense -L** command to view license information for the cluster.

Nodes with no disks in the file system are called as diskless nodes. Run the **mmchlicense client --accept -N** command to accept the client license for disks that have no disks in the GPFS file system.

Start the GPFS cluster to verify whether it starts successfully. Use the **mmstartup** *–a* command to start the GPFS cluster and the **mmgetstate** *–a* command to view the state of the GPFS cluster.

Create GPFS Network Shared Disks (NSD)

To create the network shared disks (NSD) in GPFS, create a disk file to be used as input to the **mmcrnsd** command. The disk file defines the GPFS pools and the NSDs. A recommended GPFS pool configuration has two storage pools, a system pool for metadata only and a data pool.

- Storage Pools
  - System pool contains all of the metadata disks and does not have FPO behavior enabled. The system pool should have a smaller block size than the data pool for performance reasons. If you choose to use **dataAndMetadata** disks in the system pool, you must set the system pool block size to be the same as the data pool block size as both the pools can have data. For the **dataAndMetadata** system pool, the block size 1M is recommended.
  - Data pool contains all of the data disks and has FPO behavior enabled by setting **allowWriteAffinity**=yes, **writeAffinityDepth**=1, and **blockGroupFactor**=128.

    The chunk size can be calculated as **blockSize * blockGroupFactor**. Similar to the HDFS recommendation, the GPFS FPO recommendation is **blockSize=2M * blockGroupFactor=64** for a chunk size of 128 MB
- NSD
  - Every local disk to be used by GPFS must have a matching entry in the disk stanza file
  - The device must match the device path of the disk.

    Note: In the example, /dev/sda is not included because this is the OS disk.

    If **MapReduce** intermediate and temporary data is stored on etx3/ext4 disks instead of GPFS, make sure those disks are not included in the disk file or GPFS will format them and include them in the GPFS cluster
  - System pool disks:
    - Should have **usage=metadataOnly**. It is possible to use **usage=dataAndMetadata** if there is a reason to have data on the system pool disks. The block size of the dataAndMetadata system pool must be the same as the block size of a data pool in the file system.
    - **failureGroup** must be a single number if **allowWriteAffinity** is not enabled (specify **allowWriteAffinity=no** for system pool definition when doing **mmcrnsd** or **mmcrfs**) and it should be the same for all disks on the same node. If **allowWriteAffinity** is enabled for system pool, the failure group can be of format *rack,position,node*, for example, **2,0,1**; or, it can take the traditional single-number failure group format also.
    - Even when **allowWriteAffinity** is enabled for system pool, the metadata does not follow data locality rules; these rules apply only to data placement
  - Data pool disks:

- Must have **usage**=dataOnly.
- **failureGroup** must be of the format [rack,position,node], where position is either 0 or 1 to represent top or bottom half of the rack. The sample environment does not have half racks, so the same position is used for all nodes. Especially, when position and node fields are ignored in the cluster, the failure group can be defined as a single number, [rack, -,-].

Example of NSD disk file created by using the **mmcrnsd** command:

```
%pool: pool=system  blockSize=256K layoutMap=cluster allowWriteAffinity=no
%pool: pool=datapool blockSize=2M layoutMap=cluster allowWriteAffinity=yes writeAffinityDepth=1
blockGroupFactor=256

# gpfstest9
%nsd: nsd=node9_meta_sdb  device=/dev/sdb servers=gpfstest9 usage=metadataOnly failureGroup=1 pool=system

%nsd: nsd=node9_data_sdf2  device=/dev/sdf servers=gpfstest9 usage=dataOnly failureGroup=1,0,1 pool=datapool
%nsd: nsd=node9_data_sdg2  device=/dev/sdg servers=gpfstest9 usage=dataOnly failureGroup=1,0,1 pool=datapool

#gpfstest10
%nsd: nsd=node10_meta_sda  device=/dev/sda servers=gpfstest10 usage=metadataOnly failureGroup=2 pool=system

%nsd: nsd=node10_data_sde2  device=/dev/sde servers=gpfstest10 usage=dataOnly failureGroup=2,0,1 pool=datapool
%nsd: nsd=node10_data_sdg2  device=/dev/sdg servers=gpfstest10 usage=dataOnly failureGroup=2,0,1 pool=datapool

#gpfstest11

%nsd: nsd=node11_meta_sdb  device=/dev/sdb servers=gpfstest11 usage=metadataOnly failureGroup=3 pool=system

%nsd: nsd=node11_data_sdf2  device=/dev/sdf servers=gpfstest11 usage=dataOnly failureGroup=3,0,1 pool=datapool
%nsd: nsd=node11_data_sdg2  device=/dev/sdg servers=gpfstest11 usage=dataOnly failureGroup=3,0,1 pool=datapool
```

If any disks are previously used by GPFS, you must use the `-v no` flag to force GPFS to use them again.

Note: Use the **-v no** flag only if you are sure that the disk can be used by GPFS.
Use the `# mmcrnsd -F diskfile [-v no]` command to create NSDs and use the **mmlsnsd –m** command to display the NSDs.

Apply GPFS FPO configuration changes

GPFS FPO requires several global GPFS configuration changes to operate successfully.

Set the GPFS page pool to 25% of system memory on each node. For Hadoop noSQL application, the page pool of GPFS FPO can be configured for better performance, for example, 30% of physical memory.

In this example, all nodes have the same amount of memory, which is a best practice. If some nodes have different memory, set the page pool on a per-node basis by using the `-N` flag.

```
# TOTAL_MEM=$(cat /proc/meminfo | grep MemTotal | tr -d \"[:alpha:]\" | tr -d\"[:punct:]\" | tr -d \"[:blank:]\")
# PAGE_POOL=$((${TOTAL_MEM}*25/(100*1024)))
# mmchconfig pagepool=${PAGE_POOL}M
```

Start the GPFS cluster:

```
# mmstartup -a
# mmgetstate -a
```

Use the **mmlsconfig** and **mmdiag** commands to see the configuration changes

```
# mmlsconfig
# mmdiag –config
```

Create the GPFS file system and pools

After you create NSDs, a GPFS file system can be created.

To use FPO, a single file system is recommended. The following example creates a file system with mount point **/mnt/gpfs** that is set to auto mount. This mount point is used in Hadoop configuration later. The replication for both data and metadata is set to 3 replicas. Quotas are not activated on this file system. An inode size of 4096 is recommended for typical **MapReduce**data sizes **-S** and **-E** settings help improve performance for **mtime** and **atime** updates. The **mmcrfs** command also creates GPFS storage pools based on the disk file %pools setting.

```
# mmcrfs gpfs-fpo-fs -F diskfile -T /mnt/gpfs -n 32 -m 3 -M 3 -r 3 -R 3 -i 4096 -A yes -Q no -S relatime -E no [-v no]
```

For more information on the pool configuration, see “Create GPFS Network Shared Disks (NSD)” on page 294.

Mount the file system on all nodes:

```
# mmmount all -a
```

Use the **mmlsfs** command to display the file system configuration:

```
# mmlsfs all
```

Use the **mmlsdisk** command to display the status of the NSDs:

```
# mmlsdisk gpfs-fpo-fs –L
```

Use the **mmdf** command to view the disk usage for the file system:

```
# mmdf gpfs-fpo-fs
```

Use the **mmlspool** command to view the storage pools:

```
# mmlspool gpfs-fpo-fs all –L
```

Create GPFS Data Placement Policy

Before data can be written to a GPFS file system that has more than one pool, you must apply a data placement policy.

In this example, all of the data goes to data pool.

```
# cat policyfile
rule default SET POOL 'datapool'
```

After you create the rule file, use the **mmchpolicy** command to enable the policy:

```
# mmchpolicy gpfs-fpo-fs policyfile -I yes
```

Use the **mmlspolicy** command to display the currently active rule definition:

```
# mmlspolicy gpfs-fpo-fs –L
```

Create file sets for MapReduce intermediate and temporary data

To efficiently store MapReduce intermediate and temporary data, use GPFS file sets and policies to better emulate local disk behavior.

Note: If **MapReduce** intermediate and temporary data is not stored on GPFS, **mapred.cluster.local.dir** in MRv1 or **yarn.nodemanager.log-dirs** and **yarn.nodemanager.local-dirs** in Hadoop Yarn does not point to a GPFS directory, you do not need to go through this section.

Create an independent file set

Consider using **--inode-space new [--inode-limit MaxNumInodes[:NumInodesToPreallocate]** to create an independent file set. This can improve the performance for the file set but requires calculation for **MaxNumInodes** and **NumInodesToPreallocate**. **MaxNumInodes** must be eight times the number of files expected on the file set, and **NumInodesToPreallocate** must be half the value of **MaxNumInodes**. See the **mmcrfileset** man page to understand this option.

Use the **mmcrfileset** command to create two file sets, one for local intermediate data and one for temporary data:

```
# mmcrfileset gpfs-fpo-fs mapred-local-fileset
# mmcrfileset gpfs-fpo-fs mapred-tmp-fileset
```

After the file set is created, it must be linked to a directory under this GPFS file system mount point. This example uses **/mnt/gpfs/mapred/local** for intermediate data and **/mnt/gpfs/tmp** for temporary data. As **/mnt/gpfs/mapred/local** is a nested directory, the directory structure must exist before linking the file set. These two directories are required for configuring Hadoop.

```
# mkdir -p $(dirname /mnt/gpfs/mapred/local)
# mmlinkfileset gpfs-fpo-fs mapred-local-fileset -J /mnt/gpfs/mapred/local
# mmlinkfileset gpfs-fpo-fs mapred-tmp-fileset -J /mnt/gpfs/tmp
```

Use the **mmlsfileset** command to display file set information:

```
# mmlsfileset gpfs-fpo-fs -L
```

The next step to setting up the file sets is to apply a GPFS policy so the file sets act like local directories on each node. This policy instructs GPFS not to replicate the data for these two file sets, and since these file sets are stored on the data pool, they can use FPO features that keeps local writes on local disks. Metadata must still be replicated three times, which can result in performance overhead. File placement policies are evaluated in the order they are entered, so ensure that the policies for the file sets appear before the default rule.

```
# cat policyfile
rule 'R1' SET POOL 'datapool' REPLICATE (1,1) FOR FILESET ('mapred-local-fileset')
rule 'R2' SET POOL 'datapool' REPLICATE (1,1) FOR FILESET ('mapred-tmp-fileset')
rule default SET POOL 'datapool'
# mmchpolicy gpfs-fpo-fs policyfile -I yes
```

Use the **mmlspolicy** command to display the currently active rule definition:

```
# mmlspolicy gpfs-fpo-fs –L
```

In each of these file sets, create a subdirectory for each node that run Hadoop jobs. Based on the sample environment, this script creates these subdirectories:

```
# cat mk_gpfs_local_dirs.sh
#!/bin/sh for nodename in $(mmlsnode -N all); do
 mkdir -p /mnt/gpfs/tmp/${nodename}
 mkdir -p /mnt/gpfs/mapred/local/${nodename}
 done
```

After that, on **${nodename}, link /mnt/gpfs/tmp/${nodename} /hadoop/tmp; link /mnt/gpfs/mapred/local/${nodename} /hadoop/local**. Then, in Hadoop cluster, configure **/hadoop/tmp** as **hadoop.tmp.dir** in all Hadoop nodes; configure **/hadoop/local** as **mapred.cluster.local.dir** in MRv1 or **yarn.nodemanager.log-dirs** and **yarn.nodemanager.local-dirs** in Hadoop Yarn for Hadoop nodes.

To check that the rules are working properly, you can write some test files and verify their replication settings. For example:

Create some files:

```
# echo "test" > /mnt/gpfs/mapred/local/testRep1
# echo "test" > /mnt/gpfs/testRep3
```

Use the **mmlsattr** command to check the replication settings

```
# mmlsattr /mnt/gpfs/mapred/local/testRep1
```

```
replication factors
metadata(max) data(max) file [flags]
 ------------------------------------
 1 ( 3) 1 ( 3) /mnt/gpfs/mapred/local/testRep1
 # mmlsattr /mnt/gpfs/testRep3
replication factors
metadata(max) data(max) file [flags]
------------------------------------
3 ( 3) 3 ( 3) /mnt/gpfs/testRep3
```

Set file system permissions

Depending on how different users interact with GPFS, you must create a user directory with permissions that allow users to create their own home directories.

```
# mkdir -p /mnt/gpfs/user
# chmod 1777 /mnt/gpfs/user
```

To make sure that `MapReduce` jobs can write to the GPFS file system, assign permissions to the CLUSTERADMIN user. CLUSTERADMIN is the user who starts Hadoop **`namenode`** and **`datanode`** service, for example, user hdfs.

```
# chown -R CLUSTERADMIN:CLUSTERADMINGROUP /mnt/gpfs
# chmod -R +rx /mnt/gpfs
```

Use the **`ls`** command to verify the permission settings:

```
# ls -lR /mnt/gpfs
```

Tuning your operating system for IBM Spectrum Scale

Regardless of the Hadoop distribution being used, review the following information and take the necessary action to update your configuration:

For IBM AIX® and Linux platform-specific considerations, see *Configuring and tuning your system for GPFS* in *IBM Spectrum Scale: Administration and Programming Reference*.

To avoid any potential known issues, see the *Configuration and tuning questions* section in the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Ensure the IO scheduler for the disks used by GPFS is set to **`deadline`**. You can use the following command to change disk IO scheduler online for all disks used by GPFS:

```
echo "deadline" > /sys/block/<diskname>/queue/scheduler
```

If you want the change to survive reboot, you need to enter the command under the `/etc/rc.local` folder.

Tuning IBM Spectrum Scale configuration for FPO

FPO-enabled clusters have a different architecture than traditional IBM Spectrum Scale deployments. Therefore, many of the default configuration parameters that are not suitable for FPO deployments must be modified.

Use the Table 45 on page 299 as a guide to update your cluster configuration settings.

Use the **mmchconfig** and **mmlsconfig** commands to change and list the current value of any parameter. In the following example, **readReplicaPolicy** parameter is changed to the local setting. The example also shows how to specific multiple configuration parameters in a single command.

```
#shows current setting of readReplicaPolicy parameter
mmlsconfig |grep -i readReplicaPolicy
#changes the value to policy. Use -i ensure change immediate and persistent across node reboots.
 mmchconfig readReplicaPolicy=local -i
mmchconfig maxStatCache=100000,maxFilesToCache=100000
```

To replace the parameter name and values as required run these commands on one node in the cluster, and IBM Spectrum Scale propagates the changes to all other nodes.

Note: Some parameters such as **maxStartCache** and **maxFilesToCache** do not take effect until IBM Spectrum Scale is restarted. IBM Spectrum Scale can be restarted by using the following command.

```
/usr/lpp/mmfs/bin/mmumount all -a
/usr/lpp/mmfs/bin/mmshutdown -a
/usr/lpp/mmfs/bin/mmstartup -a
#Ensure GPFS daemon has started on all nodes in the GPFS cluster.
mmgetstate -a
```

You can specify multiple configuration parameters in a single **mmchconfig** command by using comma to separate each configuration. For example, the following command can be used to set most of the parameters specified in Table 46 on page 302:

```
mmchconfig
   readReplicaPolicy=local,restripeOnDiskFailure=yes,syncBuffsPerIteration=1,minMissedPingTimeout=60,
   leaseRecoveryWait=65,prefetchAggressivenessRead=2,prefetchAggressivenessWrite=0,
   maxFilesToCache=100000,maxStatCache=100000,worker1Threads=72,nsdMinWorkerThreads=48,
   nsdInlineWriteMax=1M,nsdSmallThreadRatio=2,nsdThreadsPerQueue=10,forceLogWriteOnFdatasync=no,
   disableInodeUpdateOnFdatasync=yes,unmountOnDiskFail=meta,dataDiskCacheProtectionMethod=2
```

Table 45. IBM Spectrum Scale configuration parameter

Parameter Name	Default Value	New Value	Comment
readReplicaPolicy	Random	local	Enables the policy to read replicas from local disks.
restripeOnDiskFailure	No	Yes	Specifies whether IBM Spectrum Scale attempts to automatically recover from certain common disk failures.
metadataDiskWaitTimeForRecovery	2,400 seconds	<see comments to customize>	Sets delay period before start of recovery for failed metadata disks. Used when **RestripeOnDiskFailure** is set to yes. Ensure that the delay period is large enough to cover reboot time of the nodes hosting metadata disk servers.

Table 45. IBM Spectrum Scale configuration parameter (continued)

Parameter Name	Default Value	New Value	Comment
dataDiskWaitTimeForRecovery	3,600 seconds	<see comments to customize>	Sets delay period before start of recovery for failed data disks. Used when **RestripeOnDiskFailure** is set to yes. Datadisks are controlled through a separate tunable due to their distribution across the variety of node types. Ensure that the delay period is large enough to cover the reboot of the slowest node in the cluster.
syncBuffsPerIteration	100	1	Used to expedite buffer flush and the rename operations done by **MapReduce** jobs.
minMissedPingTimeout	3 (seconds)	10-60 (seconds)	Sets the lower bound on a missed ping timeout. For FPO clusters, a longer grace time is desirable before marking a node as dead, as it impacts all associated disks. Additionally, when running **MapReduce** workloads, the CPU can become overly busy and cause delayed ping responses. However, a longer timeout implies delay in recovery. A value between 10– 60 seconds is recommended. This value generally provides a good balance between the time to detect the real failures and the rate of false failure detection triggered by a delayed ping response due to CPU or network overload.
leaseRecoveryWait	35	65	Allows a larger grace window before starting the recovery.
pagepool	Varied	25% of	Sets the amount of physical memory that is reserved for cache on a node (use **-N** to list nodes that apply).

Table 45. IBM Spectrum Scale configuration parameter (continued)

Parameter Name	Default Value	New Value	Comment
prefetchPct	20(% of page pool)	See comments	Used by IBM Spectrum Scale as a guideline to limit the page pool space used for prefetch or write-behind buffers. For **MapReduce** workloads, sequential read and write, increase this parameter up to its 60% maximum.
prefetchThreads	72	See Comments	Controls the maximum possible threads that are dedicated to prefetching data for sequential file reads or to handle sequential write-behind. Prefetch threads must have twice the number of disksavailable to the node. Default must work well in most configurations.
maxFilesToCache	4,000	100,000	Specifies the number of I nodes to cache. Storing the inode of a file in cache permits faster reaccess to the file when retrieving location information for data blocks. Increasing this number can improve throughput for workloads with high file reuse and Hadoop **MapReduce** tasks. However, increasing this number excessively might cause paging at the file system manager node. The value must be large enough to handle the number of concurrently open files and allow caching of recently used files.

Table 46. IBM Spectrum Scale configuration parameter

Parameter Name	Default value	New value	Comment
maxStatCache	1,000	512	Specifies the number of I nodes to keep in the stat cache. The stat cache maintains only enough inode information to perform a query on the file system. **Note:** The stat cache is not effective on the Linux operating system. Therefore, you need to set the **maxStatCache** attribute to a smaller value, such as 512, on that operating system.
worker1Threads	48	See comments	Controls the maximum number of concurrent file operations at any one instant, primarily for random read and write operations that cannot be prefetched. For big data applications, increase this setting to 72.
nsdMinWorkerThreads	16	48	Increases NSD server performance by providing many dedicated threads for NSD service.
nsdInlineWriteMax	1,024	1,000,000	Defines the amount of data sent inline with write requests to the NSD server; helps to reduce overhead caused by inter-node communication.
nsdSmallThreadRatio	0	2	Changing this parameter to 2 results in assigning more resources for large I/O for Hadoop workloads.
nsdThreadsPerQueue	3	10	Shows the number of threads servicing an internal I/O queue for disk operations. Increasing this value results in increased parallelism for disk I/O and can increase throughput.

Table 46. IBM Spectrum Scale configuration parameter (continued)

Parameter Name	Default value	New value	Comment
forceLogWriteOnFdatasync	Yes	No	Controls forcing log writes to disk. When set to no, the IBM Spectrum Scale log record is flushed only when a new block is allocated to the file.
disableInodeUpdateOnFdatasync	No	Yes	When set to **yes**, the inode object is not updated on disk for mtime and atime updates on **fdatasync()** calls. File size updates are always synced to the disk.
unmountOnDiskFail	No	meta	Controls how the IBM Spectrum Scale daemon responds when a disk failure is detected. When set to **meta**, the file system is unmounted when metadata is no longer accessible. With replication factor set to 3, three failure groups must have at least one disk each in the failed state before a file system is unmounted
cnfsReboot	Yes	No	CNFS is not supported since the IBM Spectrum Scale 4.1.1 release. Use CES for functions similar to that of CNFS.
cnfsReboot	No	Yes	The **enforceFilesetQuotaOnRoot** parameter controls whether fileset quota must limit root user as any other users. The default value is No. If this parameter is set, the fileset quota is enforced on root

Table 46. IBM Spectrum Scale configuration parameter (continued)

Parameter Name	Default value	New value	Comment
dataDiskCacheProtectionMethod	0	2	Defines the kind of disks used for the GPFS file system. The default 0 indicates that disks are **powerProtected** and no recovery is needed beyond standard GPFS log recovery. A value of 2 ensures that when a node goes down, the data lost in the disk's cache is rebuilt by the GPFS. **Note:** If the physical disk-write cache is enabled, upgrade IBM Spectrum Scale to 4.1.1.2 or later and set this configuration to 2 with auto recovery enabled.

The IBM Spectrum Scale cluster and file system needs to be set up using the planning and best practices described in the "Configuring IBM Spectrum Scale Clusters" on page 293. IBM BigInsights must be installed before configuring the IBM Spectrum Scale Hadoop connector and Hadoop applications. IBM Platform Symphony can be installed after the IBM Spectrum Scale file system is created.

Monitoring and administering IBM Spectrum Scale FPO clusters

IBM Spectrum Scale supports the use of the simple network management protocol (SNMP) for monitoring the status and configuration of the IBM Spectrum Scale cluster. By using an SNMP application, the system administrator can get a detailed view of the system and receive instant notification of important events, such as a node or disk failure.

The SNMP agent software consists of a master agent and a set of subagents, which communicate with the master agent through an agent/subagent protocol, the AgentX protocol in this case.

The SNMP subagent runs on a collector node of the IBM Spectrum Scale cluster. The collector node is designated by the system administrator by using the **mmchnode** command.

The Net-SNMP master agent, also called as the SNMP daemon, or `snmpd`, must be installed on the collector node to communicate with the IBM Spectrum Scale subagent and with your SNMP management application. Net-SNMP is included in most Linux distributions and must be supported by your Linux vendor.

Refer to the GPFS SNMP support topic in the *IBM Spectrum Scale: Advanced Administration Guide* for further information about enabling SNMP support.

If using IBM BigInsights

When you install IBM Spectrum Scale, you can enable IBM Spectrum Scale monitoring using the IBM BigInsights installation program. If the monitoring was not enabled at the time of installation, it can be done later by installing the Net-SNMP master agent on the collector node to communicate with the IBM

Spectrum Scale subagent and the IBM BigInsights Console. Detailed instructions are provided in the Enabling monitoring for GPFS topic in the IBM InfoSphere® BigInsights Version 2.1.2 documentation.

If using Platform Symphony or Hadoop distribution from other vendor

You can leverage IBM Spectrum Scale SNMP integration for centralized monitoring of the IBM Spectrum Scale cluster. Follow the procedure outlined in IBM Spectrum Scale documentation under GPFS SNMP Support.

Node failure and recovery

In FPO environments with local disks, configuration option **restripeOnDiskFailure** must be set to enable automated recovery when disks become unavailable due to node outages or disk failures. When automated recovery is enabled, IBM Spectrum Scale attempts to maintain the required number of replicas on the available disks by restriping data off the failed disks—if the failure lasts longer than the period specified by the **metadataDiskWaitTimeForRecovery** and **dataDiskWaitTimeForRecovery** parameters. To avoid unnecessary data movement during a node reboot, ensure that these parameters are large enough to contain the node reboot time. See Table 2 for suggested starting values of these parameters.

When disks cannot be accessed, they are placed in a `down` state. The disk state can be seen by using the **mmlsdisk** *Device DiskName* **-L** command. If the disk becomes accessible again within the time period specified by **metadataDiskWaitTimeForRecovery** and **dataDiskWaitTimeForRecovery** parameters, the disks are started automatically by IBM Spectrum Scale. However, if disks stay in a `down` state longer than the specified period, the disk state changes to `suspended`. This causes IBM Spectrum Scale to stop allocating any new blocks on these disks and triggering the transfer of existing blocks to disks in other available failure groups.

If the failure lasts longer than the time specified by using the recovery wait parameters and if the remaining number of failure groups is not sufficient to create the specified number of replicas, files with new data blocks are marked as 'illreplicated'. You can see this marking by using the **mmlsattr –L <filename>** command.

When recovering from a node failure, you must ensure that all disks have returned to `ready` state by using the **mmchdisk** command. If the disks are marked as `suspended` due to an extended failure period, you might run the **mmchdisk <fsname> resume -a** command to return affected disks to `ready` state.

When affected disks are recovered, ill-replicated files containing the data written during the failure must be fixed by the administrator using the **mmrestripefs** or **mmrestripefile** command. The **mmrestripefs** or **mmrestripefile** command restores proper replication so that the files survive through subsequent failures. When using the **mmrestripefile** command with a file system that has a large number of files, you can use a policy rule to generate a list of ill-replicated files and run restripe to ensure full replication.

```
Policy example to generate a list of ill-replicated files.
[root@c150f1ap07 cl1]# cat illRepl.pol
RULE EXTERNAL LIST 'missRepl' EXEC ' /usr/lpp/mmfs/samples/ilm/mmpolicyExec-list.sample'
RULE 'missReplRule' LIST 'missRepl' where MISC_ ATTRIBUTES LIKE '%J%'
[root@c150f1ap07 cl1]# mmapplypolicy /gpfs/bigfs/
ingest -P ./illRepl.pol
```

Note: The sample program **/usr/lpp/mmfs/samples/ ilm/mmpolicyExec-list.sample'** needs to be modified per your needs to produce a list of files matching the rules. This program needs to be executable and available on all nodes in the cluster used by the **mmapplypolicy** command **'-N'** parameter

The file produced by the **mmapplypolicy** command is used to restore replication level of these ill-replicated files

Disk failure and recovery

Individual disk failures are handled in the same way as disk failures caused by a node failure.

If IBM Spectrum Scale detects an access problem with a specific disk, the disk is first marked `down`. Then, if the disk does not recover within the specified recovery wait time, it moves to the `suspended` state. Follow the same recovery procedure described in the node failure and recovery section.

Ingesting data into IBM Spectrum Scale clusters

`MapReduce` tasks perform best when input data is evenly distributed across cluster nodes. You can use the following approaches or a combination to ingest data for the first time and on an ongoing basis:

- Import data through a diskless IBM Spectrum Scale node. This ensures that the data is distributed evenly across all failure groups and all nodes within a failure group.
- If you have a large set of data to copy, it might help to use all cluster nodes to share ingest workload. Use a **write-affinity** depth of 0, along with as many cluster nodes with storage as possible to copy data in parallel.
- A **write-affinity** depth of 0 ensures that each node distributes data across as many nodes as possible. IBM Spectrum Scale policies can be used to enforce write-affinity depth settings based on fileset name, filename, or other attributes.
- Another mechanism to distribute data on ingest is to use **write-affinity depth failure** groups (WADFG) to control placement of the file replica. A WADFG setting of "*,*,*" ensures that all the file chunks are evenly distributed across all nodes. A placement policy can be used to selectively specify this attribute on the data set being ingested.

It is possible that even after employing the above techniques to ingest, the cluster might become unbalanced as nodes and disks are added or removed. You can check whether the data in the cluster is balanced by using the **mmdf** command. If data disks in different nodes are showing uneven disk usage, rebalance the cluster by running the **mmrestripefs -b** command. Keep in mind that the rebalancing command causes additional I/O activity in the cluster. Therefore, plan to run it at a time when workload is light.

Exporting data out of IBM Spectrum Scale clusters

In many applications, it might be required to export the output data into another system or application for further use. Hadoop native components such as Flume can be used for this purpose. If HDFS Transparency is used for Hadoop applications, distcp feature is supported over IBM Spectrum Scale to export data into a remote HDFS file system or IBM Spectrum Scale file system. Additionally, since IBM Spectrum Scale provides POSIX semantics, custom scripts can be written to move data from an IBM Spectrum Scale cluster to any other POSIX-compliant file system. Consider using CNFS to copy the needed data directly.

If data must be exported into another IBM Spectrum Scale cluster, the AFM function can be used to replicate data into a remote IBM Spectrum Scale cluster.

Upgrading FPO

When the application that runs over the cluster can be stopped, you can shut down the entire GPFS cluster and upgrade FPO. However, if the application over the cluster cannot be stopped, you need to take the rolling-upgrade procedure to upgrade nodes.

During this kind of upgrade, the service is interrupted. In production cluster, service interrupt is not accepted by the customers. If such cases, you need to take the rolling upgrade to upgrade node by node (or failure group by failure group) while keeping GPFS service up in the other nodes.

The guide for rolling upgrade is as follows:

- Only upgrade nodes from the same failure group at the same time slot; not operate nodes from two or more failure groups because bringing nodes from more than 1 failure groups will make your data exposed in data lost risk.
- Not break the quorum relationship when bringing down the nodes from one failure group. Before you bring down the nodes in one failure group, you need to check the quorum node. If bringing the quorum node in the to-be-operated failure group will break the quorum relationship in the cluster, you need to exclude that node for the rolling upgrade of the failure group.

Prerequisites

- Ensure that all disks are in a `ready` status and `up` availability. You can check by issuing the **mmlsdisk fs-name -L** command.
- Verify whether the upgraded-to GPFS version is compatible with the running version from IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html). For example, you cannot upgrade GPFS from 3.4.0.x directly into 3.5.0.24. You need to upgrade to 3.5.0.0 first and then upgrade to the latest PTF. You also need to verify whether the operating system kernel version and the Linux distro version are compatible with GPFS from IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
- Find a time period when the whole system work load is low or reserve a maintenance time window to do the upgrade. When cluster manager or file system manager is down intentionally or accidentally, another node is elected to take the management role. But it takes time to keep the cluster configuration and the file system data consistent.
- When a file system manager is elected by cluster manager, it does not change even if the file system is unmounted in this node. If the file system is mounted in other nodes, it is also 'internal' mounted in the file system manager. This does not affect your ability to unload the kernel modules and upgrade GPFS without a reboot.

Upgrade FPO as follows:

1. Disable auto recovery for disk failure

   To do a rolling upgrade of GPFS, you must shut down GPFS in some nodes. Disks in those nodes are unreachable during that time. It is better to handle this disk down manually with the following step.

   Run **`mmchconfig restripeOnDiskFailure=no -i`** in any node in cluster to disable auto recovery for disk failure. It is necessary to include **`-i`** option for this change to take effect immediately and permanently. GPFS **`restripeOnDiskFailure`** is a cluster-wide configuration. So you need to run it only once in any one node in your cluster.
2. Get the list of nodes for this upgrade cycle

   Each upgrade cycle, you can only upgrade GPFS in nodes whose disks in it have the same first two numbers in failure group vector. Save node list in file **`nodeList`**, one node name per line in it. For general information on how to specify node names, see GPFS document: Administration and Programming Reference topic: Specifying nodes as input to GPFS commands.
3. Get disk list in nodes for this upgrade cycle

   Get all disks attached in nodes for this upgrade cycle. Save it in file **`diskList`**. Each line in file **`diskList`** saves a disk name. **`mmlsdisk –L`** command can show which disks belong to the nodes you want to upgrade in this cycle.
4. Stop applications by using GPFS file system

   Confirming that there is no application which still opens file in GPFS file system. You can use **`lsof`** or **`fuse`** command to check whether there is still an open instance active for file in GPFS file system.
5. Unmount GPFS file system

   Unmount GPFS file system in all nodes for this upgrade cycle through command:

   ```
   mmumount <fsName> -N <nodeList>
   ```

Confirming file system has already unmounted in all related nodes through command:

```
mmlsmount <fsName> -L
```

6. Suspend disks

   Suspend all attached disks in nodes for this upgrade cycle to make sure that GPFS does not try to allocate new data block from these disks. GPFS can still read valid data block from suspended disk

   ```
   mmchdisk <fsName> suspend -d <diskList>
   ```

   Confirming disks are suspended properly through command:

   ```
   mmlsdisk <fsName>
   ```

7. Shut down GPFS

   Shut down GPFS in all nodes for this upgrade cycle through command:

   ```
   mmshutdown -N <nodeList>
   ```

   Confirming GPFS is in `down` status in these nodes through command:

   ```
   mmgetstate -a
   ```

8. Upgrade GPFS

   You can upgrade GPFS packages in each node of this upgrade cycle. For general information on how to install GPFS packages on nodes, see the GPFS: Administration and Programming Reference topic: Installing GPFS on Linux/AIX/Windows nodes.

9. Start GPFS

   After all packages are ready, you can start up GPFS:

   ```
   mmstartup -N <nodeList>
   ```

   Confirming GPFS are in "Active" status in these nodes through command:

   ```
   mmgetstate -a
   ```

10. Resume and start disks

    When GPFS is in "Active" status in all nodes, you can resume disks which were suspended intentionally in Step 7.

    ```
    mmchdisk <fsName> resume -a or
    mmchdisk <fsName> resume -d <diskList>
    ```

    When these disks are in "ready" status and if some of these disks are in “down” availability, you can start these disks through the following command:

    ```
    mmchdisk <fsName> start -a or
    mmchdisk <fsName> start -d <diskList>
    ```

    This might take a while since GPFS must do incremental data sync up to keep all data in these suspended disks are up-to-date. The time it needs depends on how much data has changed when the disks were kept in `suspende` status. You have to wait for **`mmchdisk start`** command to finish to do next step.

    Confirming all disks are in `ready`status and `up` state through command:

    ```
    mmlsdisk <fsName>
    ```

11. Mounts GPFS file system

    When all disks in the file system are in `up` status you can mount file system:

    ```
    mmmount <fsName> -N <nodeList>
    ```

    Confirming GPFS file system is mounted properly through command:

    ```
    mmlsmount <fsName> -L
    ```

    Repeat Step 3 to Step 12 to upgrade GPFS in all nodes in your cluster.

12. Enable auto recovery for disk failure

    Now you can enable auto recovery for disk failure through command:

    ```
    mmchconfig restripeOnDiskFailure=no -i
    ```

It's necessary to includes **-i** option for this change to take effect immediately and permanently.

13. Upgrade GPFS cluster version and file system version

    Issue **mmchconfig release=LATEST** and **mmchfs -V compat** to ensure the upgrade is successful and the cluster would never revert to the old build for minor GPFS upgrade. It is recommended to use **mmchfs –V compat** to enable backward-compatible format changes.

    For major GPFS upgrade, consult IBM Support Center to verify compatibility between the different GPFS major versions, before issuing **mmchfs –V full**. For information about specific file system format and function changes when you upgrade to the current release, see the following GPFS: Administration and Programming Reference topic: Chapter 11, "File system format changes between versions of GPFS,"

Restrictions

An FPO environment includes restrictions.

The following restrictions apply:

- Storage pool properties can be set only when the pool is created and cannot be changed later.
- All disks in an FPO pool must be assigned an explicit failure group.
- All disks in an FPO pool must have exactly one NSD server associated with them.
- All disks in an FPO pool that share an NSD server must belong to the same failure group.
- When replacing a disk in an FPO pool, the old and new disks must have the same NSD server.
- Disks must be removed from the file system before NSD servers can be changed.
- FPO is not supported on Windows.

There might be additional limitations and restrictions. For the latest support information, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Chapter 18. Hadoop support for IBM Spectrum Scale

The Apache Hadoop framework features open source software that enables distributed processing of large data sets across clusters of commodity servers. It is designed to scale up from a single server to thousands of machines, with a high degree of fault tolerance.

The Apache Hadoop framework allows distributed processing of large data sets across clusters of computers that use simple programming models. The Apache Hadoop framework processes data-intensive computational tasks, which include data amounts that can range from hundreds of terabytes (TBs) to tens of petabytes (PBs). This computation mode differs from the computation mode that is used in traditional high-performance computing (HPC) environments.

For example, Hadoop consists of many open source modules. One of the primary open source modules is the Hadoop Distributed File System (HDFS), which is a distributed file system that runs on commodity hardware. HDFS lacks enterprise class capabilities necessary for reliability, data management, and data governance. IBM Spectrum Scale, which is a scale-out distributed file system, offers an enterprise-class alternative to HDFS.

Hadoop collaboration with IBM Spectrum Scale

IBM Spectrum Scale provides integration with Hadoop applications that use the Hadoop connector (so you can use IBM Spectrum Scale enterprise-level functions on Hadoop):

- POSIX-compliant APIs or the command line
- FIPS and NIST compliant data encryption
- Disaster Recovery
- Simplified data workflow across applications
- Snapshot support for point-in-time data captures
- Simplified capacity management by using IBM Spectrum Scale (for all storage needs)
- Policy-based information lifecycle management capabilities to manage PBs of data
- Infrastructure to manage multi-tenant Hadoop clusters based on service-level agreements (SLAs)
- Simplified administration and automated recovery
- Multiple clusters

Hadoop connector

This section further describes how the Hadoop connector enables Hadoop for IBM Spectrum Scale.

Hadoop connector support for IBM Spectrum Scale

The IBM Spectrum Scale Hadoop connector, which must be installed on each Hadoop node, implements Hadoop file system APIs and the `FileContext` class so it can access the IBM Spectrum Scale.

The IBM Spectrum Scale Hadoop connector is composed of these parts:

- `hadoop-gpfs-*.jar`
- `libgpfshadoop.so`
- `gpfs-connector-daemon`

Install the connector on each node in the Hadoop cluster.

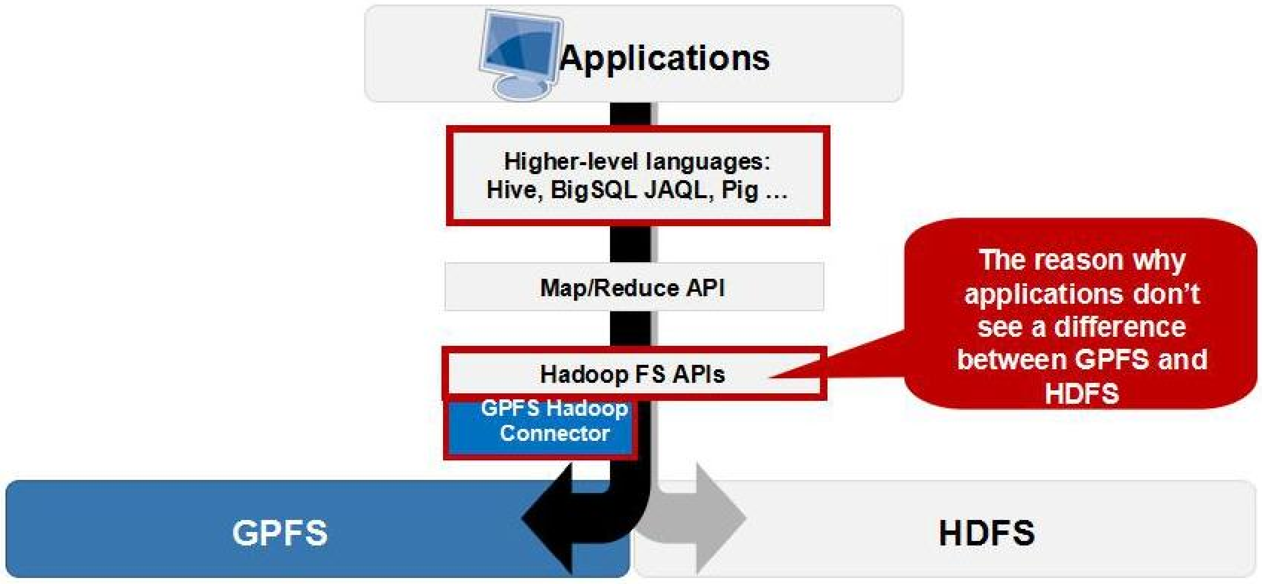

Applications
Higher-level languages: Hive, BigSQL JAQL, Pig ...
Map/Reduce API
Hadoop FS APIs
GPFS Hadoop Connector
The reason why applications don't see a difference between GPFS and HDFS
GPFS
HDFS

Figure 13. GPFS Hadoop connector overview

Note: Specify the **/usr/lpp/mmfs/bin/mmhadoopctl** command to deploy the connector on Hadoop and to control the connector daemon.

Hadoop support for the IBM Spectrum Scale storage mode

IBM Spectrum Scale has two storage modes that Hadoop can access:

- Centralized Storage Mode
- Local Storage Mode, or File Placement Optimizer (FPO) Mode

IBM Spectrum Scale allows Hadoop applications to access centralized storage data like the SAN Volume Controller, which means that storage is attached to a dedicated storage server (or accessed directly by SAN Volume Controller). All Hadoop replica nodes can access the storage as a IBM Spectrum Scale client. You can share a cluster between Hadoop and any other application.

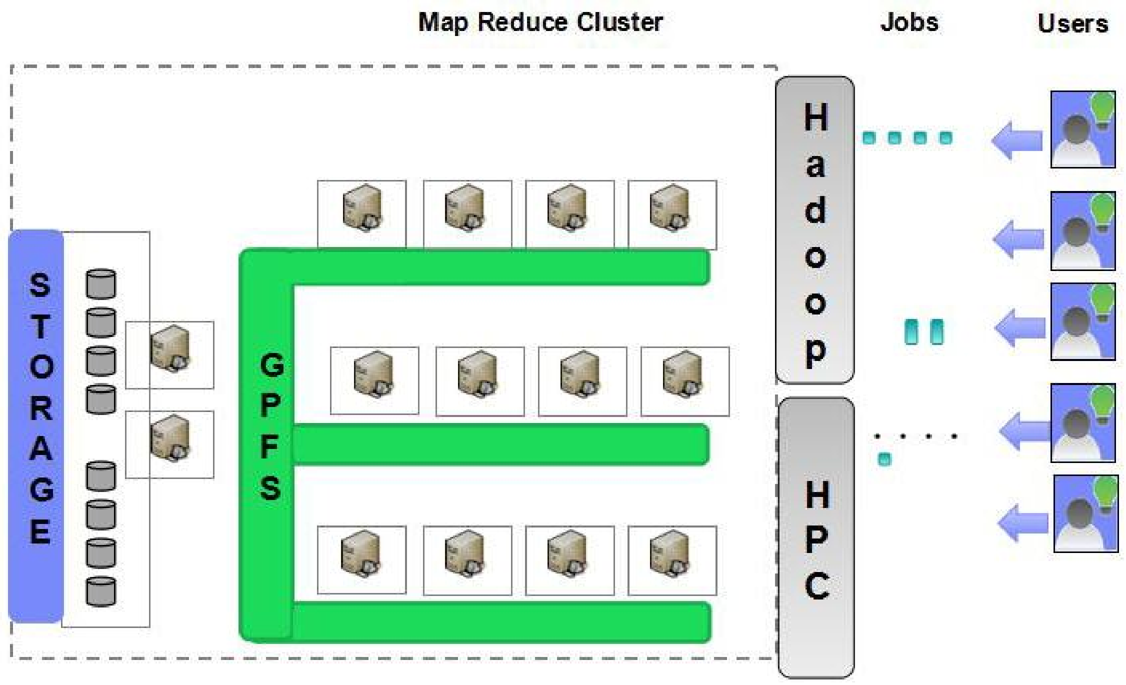


Figure 14. Hadoop on centralized storage

IBM Spectrum Scale also provides a local storage mode, which is FPO (for Hadoop). FPO is an implementation of shared-nothing architecture that enables each node to operate independently, which reduces the impact of failure events that occur across multiple nodes.

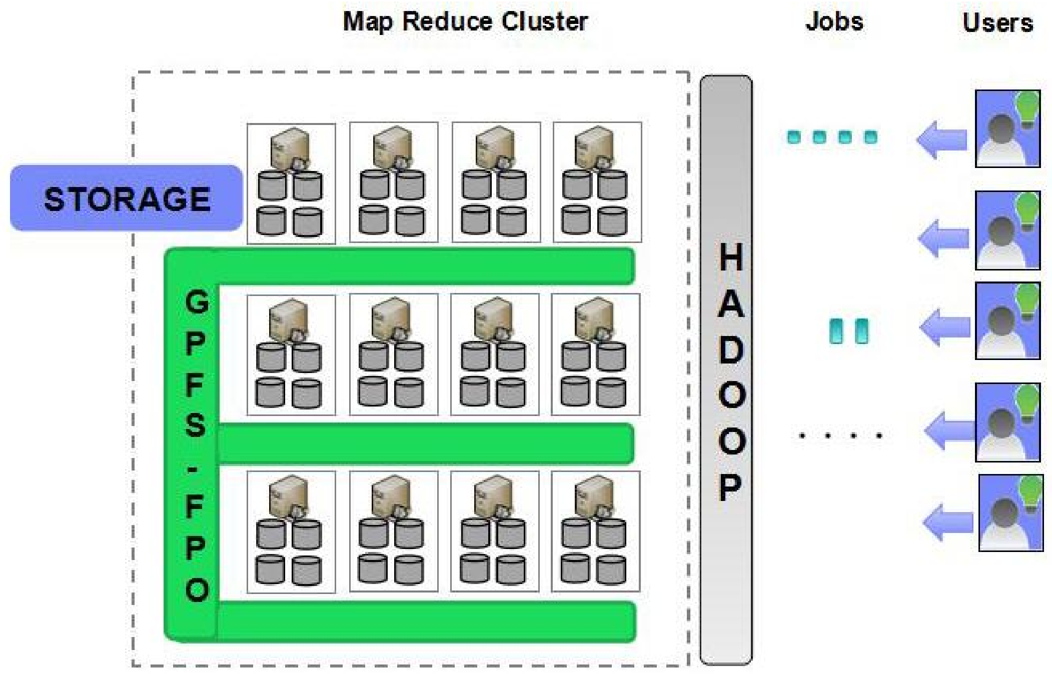


Figure 15. Hadoop on Local storage (FPO)

IBM Spectrum Scale cluster planning for Hadoop applications

When you are creating a IBM Spectrum Scale file system for a Hadoop application:

- IBM Spectrum Scale allows Hadoop to run on a file system with multiple storage pools. For purposes of storage pool planning, these storage pools can be either generic or FPO (with `allowWriteAffinity=yes`). With central storage, IBM Spectrum Scale can run Hadoop on a typical file system configuration that has a mixed storage pool with both metadata and data.
- You must consider replica and block size planning:

Table 47. Replica and block size planning

Mode	Data type	Block size	Replica
FPO mode	Metadata	A small size is suggested. For example, 256 KB.	2 or 3 Fewer replicas apply to disks with hardware protection such as RAID.
	Data	A large size is suggested. A typical value is 2 MB.	3 Replica 2 is only considered with extra disk protection such as RAID.
Central Storage mode	Metadata	It depends on the application I/O pattern and whether the application is using RAID or not.	1 or 2
	Data	It depends on the application I/O pattern and whether the application is using RAID or not.	1 or 2

- You can use any advanced features in IBM Spectrum Scale, such as Local Read-Only Cache (LROC), to improve performance.
- Hadoop can run on a remote file system that is mounted from another cluster.

Hadoop cluster planning

In an IBM Spectrum Scale shared storage environment, do not use an NSD server as a computing node because NSD servers generally maintain a heavy I/O workload.

Remember: In a Scale Central Storage (SCS) environment, any node can be used as a Hadoop replica node.

IBM Spectrum Scale functions corresponding to HDFS

To enable HDFS-4685 access control lists (ACLs):

1. Enable GPFS Posix ACL support:

```
# mmchfs bigfs -k posix
# mmlsfs bigfs -k

flag                value                    description
----                -----                    -----------
-k                  posix                    ACL semantics in effect
```

2. Install the dependent packages:
   - `acl`
   - `libacl`
   - `libacl-devel` (required by Hadoop 2.4 connector only)

In this example, the IBM Spectrum Scale file system name is `bigfs`.

To enable the HDFS-2006 ability to store extended attributes per file, install the dependent `libattr` package.

To enable the remote file system in the Hadoop connector, add the following property:

```
<property>
<name>gpfs.remote.cluster.enabled</name>
<value>true</value>
</property>
```

Installing the IBM Spectrum Scale Hadoop connector

IBM Spectrum Scale Hadoop connector 2.7 is installed under the `/usr/lpp/mmfs/hadoop` directory.

See the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) to get the list of Hadoop versions supported by IBM Spectrum Scale Hadoop connector. It is highly recommended that you use the latest version of Hadoop connector.

If you are running old Hadoop, you need to install old IBM Spectrum Scale Hadoop connector version.

Different Hadoop distributions and versions might require different methods to deploy and configure the IBM Spectrum Scale Hadoop connector. If you are using IBM BigInsights 3.0.x or any earlier release, IBM Spectrum Scale Hadoop connector can be installed and configured by IBM BigInsights automatically. If you are using IBM Platform Symphony with IBM Spectrum Scale, follow the procedure described in the IBM Platform Integration Guide for MapReduce Applications to install IBM Platform Symphony, and then configure IBM Spectrum Scale Hadoop connector by following the guide shipped with IBM Platform Symphony. If you are using IOP 4.x, you can use Ambari to install and configure IBM Spectrum Scale cluster and connector automatically. For more information, see the "Deploy IBM Spectrum Scale using Ambari" on page 325 topic. If you are using Hadoop distribution from Apache or other vendors, check your Hadoop distribution document to check the prerequisites, and then deploy the IBM Spectrum Scale and connector.

To install the IBM Spectrum Scale Hadoop 2.7 connector, download it from the IBM developerWorks® Wiki.

If you are using Hadoop-2.7 version, do the following steps to install and configuring the IBM Spectrum Scale Hadoop connector.

1. Run the following command to install IBM Spectrum Scale Hadoop connector:
   ```
   mmhadoopctl connector attach --distribution <BigInsights or Apache> -N all
   ```
2. Start IBM Spectrum Scale Hadoop connector by running the following command:
   ```
   mmhadoopctl connector start -N all
   ```
3. Check the IBM Spectrum Scale Hadoop connector health by running the following command:
   ```
   mmhadoopctl connector getstate -N all
   ```

Note: The `–N` option is only supported after IBM Spectrum Scale 4.1.1 PTF1 release and later.

See the Modifying the Hadoop configuration to use IBM Spectrum Scale topic for additional applicable configuration change to Hadoop configuration files.

Modifying the Hadoop configuration to use IBM Spectrum Scale

This section describes configuration updates to the Hadoop MapReduce applications. Regardless of the Hadoop distribution being used, you should see the settings below and make the required changes before running MapReduce applications.

The following configuration changes are for Hadoop 2.7 when taking BigInsights 4.0/4.1. When taking other Hadoop distributions, you need to check the configuration against your Hadoop distributions and update the configuration by using the interface provided by the vendor.

Hadoop component storage falls into two main categories:

- **Shared distributed storage**: Referred to as the Distributed File System in Hadoop documentation and configuration files, these paths map to the IBM Spectrum Scale file system. By default, Hadoop configuration files are defined to use HDFS and need to be updated as described below.
- **Local storage**: This term refers to the storage managed as the file system locally on the node. Paths are specified for local storage map to a local file system on the node such as `ext3`, `ext4`. Local storage is used to keep log files generated by Hadoop components and other temporary files that do not need to be distributed. Using the local storage for temporary files can provide better performance, as it avoids replication overhead for metadata and data.

Note: If you are using IBM BigInsights, all changes must be made through IBM BigInsights Web GUI instead of modifying configuration file directly. IBM BigInsights uses a database to manage Hadoop configurations. Hadoop configuration files would be refreshed when IBM BigInsights services are restarted. You must stop services before applying the below changes and then start service for the changes to take effect.

The `core-site.xml` file contains the site-specific configuration for a given Hadoop installation.

You must stop HDFS services before applying the below changes and start HDFS service for the changes to take effect, and then stop HDFS server again. For HDFS service, check the Summary section to ensure HDFS related services are down. Ignore any failure message here since this step needs to only save changes into IBM BigInsights repository and configuration files.

Table 48. Modifications to Hadoop configuration—core-site.xml

Name	Value	Notes
`fs.hdfs.impl`	See notes	This configuration specifies the Spectrum Scale connector class name. Add the configuration through web GUI. Choose **HDFS service** > **Configs**, in the **Custom core-site** section, add this new configuration and set it to `org.apache.hadoop.fs.gpfs.GeneralParallelFileSystem` value .
`fs.gpfs.impl`	See notes	This configuration specifies the IBM Spectrum Scale connector class name. Add the configuration through web GUI. Choose **HDFS service** > **Configs**, in the **Custom core-site** section, add this new configuration and set it to `org.apache.hadoop.fs.gpfs.GeneralParallelFileSystem` value .
`gpfs.mount.dir`	absolute path to IBM Spectrum Scale file system mount directory	This configuration specifies the mount point of the IBM Spectrum Scale file system on all nodes. For example, under `/gpfs/bigfs`. Add the configuration through web GUI. Choose **HDFS service** > **Configs**, in the **Custom core-site** section, add this new configuration and set it to the absolute path to local file system mount directory.

Table 48. Modifications to Hadoop configuration—core-site.xml (continued)

Name	Value	Notes
`hadoop.tmp.dir`	absolute path to local file system mount directory	This configuration specifies a path on the local file system. This is used as a base directory for local Hadoop files, task output and error files, and task-specific Java™ class. This path must exist on each node. For example, under `/hadoop/local/sd1/tmp`. Add the configuration through web GUI. Choose **HDFS service** > **Configs**, in the **Custom core-site** section, add this new configuration and set it to the absolute path to local file system mount directory.
`fs.AbstractFileSystem.hdfs.impl`	See notes	Add the configuration through web GUI. Choose **HDFS service** > **Configs**, in the **Custom core-site** section, add this new configuration and set it to the `org.apache.hadoop.fs.gpfs.GeneralParallelFs` value.
`fs.AbstractFileSystem.gpfs.impl`	See notes	Add the configuration through web GUI. Choose **HDFS service** > **Configs**, in the **Custom core-site** section, add this new configuration and set it to the `org.apache.hadoop.fs.gpfs.GeneralParallelFs` value.
`gpfs.supergroup`	hadoop	Configure the groups that are privileged on the file system. Multiple groups are separated by comma. There are two kinds of user groups in Hadoop, one is super user group while the other is normal user. The super user group must have super permission to access distributed file system. All request from super user will redirect to IBM Spectrum Scale Hadoop connector and executed as root user. All the other user's request are handled locally by JVM. Add the configuration through web GUI. Choose **HDFS service** > **Configs**, in the **Custom core-site** section, add this new configuration and set it to the specific value. **Note:** The group **`hadoop`** is the group name of all users from a different service, for example, hbase, hive, and yarn. If the customer takes a non-default group name for these different service users, then the **`gpfs.supergroup`** must be the group name used by the customer.
`gpfs.remote.cluster.enabled`	true	IBM Spectrum Scale support multiple clusters, so a file system might mount from a remote cluster. You can run Hadoop on this sort of cluster by enabling the configuration. As the local cluster cannot make use of data locality in remote filesystem, connector reports an arbitrary hostname for data block location. The hostname for this usage is **`gpfsNSDserver`** by default. Do not use this reserved name as any host name in cluster as then Hadoop locates more tasks on the **`gpfsNSDserver`** node, which makes the workload unbalanced. **Note:** This configuration is not required if the file system is not mounted remotely.

If you are using IBM Platform Symphony or another Apache Hadoop distribution, the file must be located under `<Hadoop install path>/conf/`. The file must be updated on each node.

The `mapred-site.xml` file contains configuration information that overrides the default values for **`MapReduce`** parameters.

If you are using IBM BigInsights, all changes must be made through IBM BigInsights Web GUI instead of modifying configuration file directly. You must stop **MapReduce2** services before applying following changes and start **MapReduce2** service for the changes to take effect. If you are using IBM Platform Symphony or another Apache Hadoop distribution, the file must be located under `<Hadoop install path>/conf/`. The file must be updated on each node.

Table 49. Modifications to Hadoop configuration—mapred-site.xml

Name	Value	Notes
`mapred.cluster.local.dir`	See notes	This is the local directory where MapReduce stores intermediate data files. It migth be a comma-separated list of directories on different devices in to spread disk I/O. Directories that do not exist are ignored. If multiple disk partitions were created during preparation, use those partitions by specifying a path on each local file system on the second disk partition. This configuration controls only temporary file from MapReduce V1 jobs. YARN job temporary files location are controlled by configurations mentioned in Table 50 on page 319. For example, In the `/hadoop/local/sd1/local,/hadoop/local/sd2/local,/hadoop/local/sd3/local`, add the configuration through web GUI. Choose **MapReduce2 service > Configs** in **Custom mapred-site** section. Add **mapred.cluster.local.dir** configuration and set it to the absolute path in local file system mount directory where you want to store MapReduce intermediate data files. It is not supported to save MapReduce intermediate data into the IBM Spectrum Scale file system if using IBM BigInsights. **Note:** This variable is only effective for Hadoop MR V1 jobs. For MR2, see the Yarn's configuration.
`mapreduce.map.java.opts`, `mapreduce.reduce.java.opts` and `yarn.app.mapreduce.am.command-opts`	`-Xmx1229m` (See notes)	These values specify the heap memory used by the Java Virtual Machine running MapReduce task. It must be set based on the amount of memory on each computing node after 25 percent is set aside for the IBM Spectrum Scale page pool. It also helps determine the number of tasks that can be run in parallel on each node. For example, on a node with 48 GB memory, 1,000m per task would approximately allow 18 map tasks and 3 reduce tasks. Update the configuration through web GUI. Choose **MapReduce2 service > Configs**, in the **Advanced mapred-site** section, Update `mapreduce.map.java.opts/mapreduce.reduce.java.opts/yarn.app.mapreduce.am.command-opts` if these values do not fit your system configuration.
`mapreduce.application.classpath`	See notes	Update the configuration through web GUI. Choose **MapReduce2 service > Configs**, in the **Advanced mapred-site** section,, add the string `/usr/iop/current/hadoop-client/hadoop-gpfs.jar` into the beginning of existing value.
`mapreduce.application.framework.path`	Keep the default value. See notes.	If your MapReduce job does not find frame work package, ensure IBM Spectrum Scale file system is mounted, HDFS service is down, MapReduce2 and YARN service are up, and run the following command to resolve the problem: `hadoop fs -put /usr/iop/$IOP_VERSION/hadoop/mapreduce.tar.gz /iop/apps/$IOP_VERSION/mapreduce/mapreduce.tar.gz`

For YARN, site-specific customization is done by specifying overriding parameters in the `Yarn-site.xml` file.

If using IBM BigInsights, all changes must be made through IBM BigInsights Web GUI instead of modifying configuration file directly. You must stop **Yarn** services before applying following changes and start **Yarn** service for the changes to take effect.

Table 50. Modifications to Hadoop configuration—Yarn-site.xml

Name	Value	Notes

yarn.nodemanager.local-dirs	See notes	This is the local directory where YARN stores intermediate data files. It migth be a comma-separated list of directories on different devices to spread disk I/O. Directories that do not exist are ignored. If multiple disk partitions were created during preparation, use those partitions by specifying a path on each local file system on the second disk partition. This configuration only controls temp file from YARN jobs.
		
		For example: in the `/hadoop/local/sd1/local,/hadoop/local/sd2/local,/hadoop/local/sd3/local` directory. Add the configuration through web GUI. Choose **MapReduce2** service **> Configs**, in the **Custom yarn-site** section, add the **yarn.nodemanager.local-dirs** configuration and set it to the absolute path in local file system mount directory where you want to store YARN intermediate data files. You cannot save YARN's intermediate data into the IBM Spectrum Scale file system if using IBM BigInsights.
		
		If you want to put the data into the shared storage, you need to take the following steps:
		1. Create one fileset name `local` with onereplica for less performance impact. Add the following policy for the fileset `local` into your policy file:
		`rule 'local' SET POOL 'datapool' REPLICATE (1,1) FOR FILESET (local)`
		
		You can also make the policy effective by using the **`mmchpolicy`** command.
		2. **`mmlinkfileset gpfs-fs-name local -J gpfs-mount-point/local`**
		3. `mkdir /<gpfs-mount-point>/local/<hostnameX>` for all hosts in the Hadoop cluster, for example, `/<gpfs-mount-point>/local/host1, /<gpfs-mount-point>/local/host2, /<gpfs-mount-point>/local/host3...`).
		4. On each host, run the following command:
		`ln -s /<gpfs-mount-point>/local/<hostnameX> /hadoop/local/local_dir`
		5. Configure `yarn.nodemanager.local-dir` as `/hadoop/local/local_dir` on Ambari GUI.

Table 50. Modifications to Hadoop configuration—Yarn-site.xml (continued)

Name	Value	Notes

yarn.nodemanager.log-dirs	See notes	This is the local directory where YARN stores log files.
		
		Either this can be a local directory where YARN stores log files in each node or it can be a directory from IBM Spectrum Scale file system as it is then easy to check YARN log from all nodes in one place.
		
		For example: in the `/hadoop/local/sd1/local,/hadoop/local/sd2/local,/hadoop/local/sd3/local` directory. Add the configuration through web GUI. Choose **MapReduce2** service **> Configs**, in the **Custom yarn-site** section, add the **yarn.nodemanager.logs-dirs** configuration and set it to the absolute path in local file system mount directory where you want to store YARN log files.
		
		If you want to put the data into the shared storage, you need to take the following steps:
		1. Create one fileset name `local` with one replica for less performance impact Add the following policy for the fileset `local` into your policy file:
		`rule 'local' SET POOL 'datapool' REPLICATE (1,1) FOR FILESET (local)`
		
		You can also make the policy effective by using the **`mmchpolicy`** command.
		2. **`mmlinkfileset gpfs-fs-name local -J gpfs-mount-point/local`**
		3. `mkdir /<gpfs-mount-point>/local/<hostnameX>` for all hosts in the Hadoop cluster, for example, `/<gpfs-mount-point>/local/host1, /<gpfs-mount-point>/local/host2, /<gpfs-mount-point>/local/host3...`).
		4. On each host, run the following command:
		`ln -s /<gpfs-mount-point>/local/<hostnameX> /hadoop/local/local_dir`
		5. Configure `yarn.nodemanager.local-dir` as `/hadoop/local/local_dir` on Ambari GUI.

For Hive, site-specific customization is done by specifying overriding parameters in the `hive-site.xml` file.

If using IBM BigInsights, all changes must be made through IBM BigInsights Web GUI instead of modifying configuration files directly.

You must stop the **Hive** services before applying the following changes and start the **Hive** service for the changes to take effect. If using IBM Platform Symphony or another Apache Hadoop distribution, the file must be located under the `<hive_home>/conf/` folder. The file must be updated on each node.

Table 51. Modifications to Hadoop configuration—hive-site.xml

Name	Value	Notes

`hive.exec.scratchdir`	relative path in GPFS file system	This value specifies a path in the Spectrum Scale file system to be used for temporary data. This can be defined on a fileset that has the replication factor set to 1 through policy rules. The path specified is relative to the IBM Spectrum Scale file system mount point.
		
		For example, in the `/hive-scratch` directory, update the configuration through web GUI.
		
		Choose **Hive** service **>** **Configs**, in the **Advanced hive-site** section, update this configuration to the absolute link path of the IBM Spectrum Scale fileset if you decide to use Spectrum Scale to save Hive temporary data.
		
		If you are using other user, such as hive, instead of the root user to start the **Hive** service, you need to change the owner and other related permission of this directory to specify the user. For example,
		`chown hive:hadoop /hive-scratch`
		`chmod 733 /hive-scratch`

For Ooize, site-specific customization is done by specifying overriding parameters in the `Ooize-site.xml` file.

If using IBM BigInsights, all changes must be made through IBM BigInsights Web GUI instead of modifying configuration file directly.

You must stop **Ooize** services before applying the following changes, and then start **Ooize** service for the changes to take effect.

Table 52. Modifications to Hadoop configuration—Ooize-site.xml

Name	Value	Notes
`oozie.service.HadoopAccessorService.supported.filesystems`	*	This value specifies supported file systems for Ooize services. Update the configuration through web GUI. Choose **Ooize** service **>** **Configs**, in **Custom ooize-site** section, update this configuration to value "*".

For HBase, site-specific customization is done by specifying overriding parameters in the `hbase-site.xml` file.

If using IBM BigInsights, all changes must be made through IBM BigInsights Web GUI instead of modifying configuration file directly.

You must stop **HBase** services before applying the following changes and start **HBase** service for the changes to take effect. If using IBM Platform Symphony or another Apache Hadoop distribution, the file must be located under `<hbase home>/conf/` directory. The file must be updated on each node.

Table 53. Modifications to Hadoop configuration—hbase-site.xml.

Name	**Value**	**Notes**
`gpfs.sync.queue`	`True`	Use queues to merge commit logs in to a single write instead of multiple small writes. Add the configuration through web GUI. Choose **HBase** service **> Configs**, in the **Custom hbase-site** section, add this configuration and set it to the specific value.
`gpfs.sync.range`	`True`	Use the sync-range semantics for fsync to sync the log that was written instead of using fsync for data blocks. Add the configuration through web GUI. Choose **HBase** service **> Configs**, in the **Custom hbase-site** section, add this configuration and set it to the specific value.
`hbase.fsutil.hdfs.impl`	`org.apache.hadoop.hbase.` `gpfs.util.FSGPFSUtils`	Add the configuration through web GUI. Choose **HBase** service **> Configs**, in the **Custom hbase-site** section, add this configuration and set it to the specific value.
`hbase.regionserver.hlog.writer.impl`	`org.apache.hadoop.hbase.gpfs.` `regionserver.wal.` `PreallocatedProtobufLogWriter`	Required for enhanced performance on the hbase log writes by pre-allocating space for log files. Add the configuration through web GUI. Choose **HBase** service **> Configs**, in the **Custom hbase-site** section, add this configuration and set it to the specific value.
`hbase.regionserver.hlog.reader.impl`	`org.apache.hadoop.hbase.gpfs.` `regionserver.wal.` `PreallocatedProtobufLogReader`	Required for enhanced performance on the hbase log reads by pre-allocating space for log files. Add the configuration through web GUI. Choose **HBase** service **> Configs**, in the **Custom hbase-site** section, add this configuration and set it to the specific value.

For Spark, site-specific customization is done by specifying overriding parameters in the `Spark-site.xml` file.

If using IBM BigInsights, all changes must be made through IBM BigInsights Web GUI instead of modifying the configuration file directly.

You must stop **Spark** services before applying the following changes and start **Spark** service for the changes to take effect.

Table 54. Modifications to Hadoop configuration—Spark-site.xml.

Name	Value	Notes
spark-env template	See notes	This value maintains Spark services running environment. Update the configuration through web GUI. Choose **Spark** service **> Advanced spark-env**, in the **spark-env template** section Add **`SPARK_CLASSPATH`** parameter with the value: `export SPARK_CLASSPATH= /usr/iop/current/hadoop-client/*: /usr/iop/current/hadoop-client/lib/*`
spark.sql.hive.metastore.sharedPrefixes	`org.apache.hadoop.fs.gpfs,org. apache.hadoop.hbase.gpfs,org.apache. hadoop.mapred.gpfs,org.apache.hadoop.pig.gpfs, org.apache.hadoop`	Update the configuration through web GUI. Choose **Spark**" service **> Configs > Custom spark-defaults**, add the variable with the specified value.

Propagating changes to all cluster nodes

- If you are using IBM BigInsights, after updating the applicable configurations through the web GUI, changes are propagated to the entire cluster.
- If you are using IBM Platform Symphony or another Hadoop distribution, follow the steps provided by the vendor to update all the configurations files like `core-site.xml` and `mapred-site.xml` files on all Hadoop cluster nodes.

Upgrading IBM Spectrum Scale connector

When you are upgrading the connector for one node, the Hadoop applications running over the node cannot access the data in the IBM Spectrum Scale file system, although upgrading the connector does not impact the IBM Spectrum Scale file system service over the node. The customer needs to plan upgrading the connector according to the Hadoop application service.

Upgrading IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 release, this section explains the steps required to upgrade the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

Also, you must ensure there are no two versions of `hadoop-gpfs-*.jar` and `libgpfshadoop.so` under the Hadoop distribution, to avoid exceptions when running jobs over Hadoop.

1. Remove any links or copies of the `hadoop-gpfs-2.4.jar` file from your Hadoop distribution directory. Also, remove any links or copies of the `libgpfshadoop.64.so` file from your Hadoop distribution directory.

   Note: For IBM BigInsights IOP 4.0, the distribution directory is `/usr/iop/4.0.0.0`.
2. Stop the current connector daemon:
   ```
   # ps -elf | grep gpfs-connector-daemon
   # kill -9 <pid-of-connector-daemon>
   ```
3. Run the following commands, to remove callbacks from IBM Spectrum Scale:
   ```
   cd /usr/lpp/mmfs/fpo/hadoop-2.4/install_script
   ./gpfs-callbacks.sh --delete
   ```

Run the **mmlscallbacks all** command to check whether connector-related callbacks, such as `callback ID start-connector-daemon` and `stop-connector-daemon`, are removed. The IBM Spectrum Scale Hadoop connector callbacks are cluster-wide and this step is required to be done over any one of nodes.

4. Remove the following files:

```
rm -f /var/mmfs/etc/gpfs-callbacks.sh
rm -f /var/mmfs/etc/gpfs-callback_start_connector_daemon.sh
rm -f /var/mmfs/etc/gpfs-callback_stop_connector_daemon.sh
rm -f /var/mmfs/etc/gpfs-connector-daemon
```

5. Run the following installation command:

   `–ivh gpfs.hadoop-connector-2.7.0-*.<platform-arch>.rpm`

6. **`mmhadoopctl connector attach --distribution BigInsights`**

   Note: The **mmhadoopctl** command does not display any help in IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 releases.

7. Restart the IBM BigInsights IOP services over the current operating node.

Upgrading IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.7 (efix3) or 4.1.0.8 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.7 (efix3) or 4.1.0.8 releases, this section explains the steps required to upgrade the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

Also, ensure that there are no two versions of `hadoop-gpfs-*.jar` and `libgpfshadoop.so` under the Hadoop distribution, to avoid exceptions when running jobs over Hadoop.

1. **`mmhadoopctl connector stop`**
2. **`mmhadoopctl connector detach --distribution BigInsights`**
3. Run the following installation command:

   `rpm –ivh --force gpfs.hadoop-connector-2.7.0-<your-new-build-number>.<platform-arch>.rpm`

   Note: : To avoid conflict for `/usr/lpp/mmfs/bin/mmhadoopctl`, `--force` is required in this step.
4. **`mmhadoopctl connector attach --distribution BigInsights`**
5. Restart the IBM BigInsights IOP services over the current operating node.

Upgrading IBM Spectrum Scale Hadoop connector 2.4 or 2.5 over IBM Spectrum Scale 4.1.1 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 or 2.5 in IBM Spectrum Scale 4.1.1 and later, this section explains the steps required to upgrade the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

Also, ensure that there are no two versions of `hadoop-gpfs-*.jar` and `libgpfshadoop.so` under the Hadoop distribution, to avoid exceptions when running jobs over Hadoop.

1. **`mmhadoopctl connector stop`**
2. **`mmhadoopctl connector detach --distribution BigInsights`**
3. **`rpm -e gpfs.hadoop-2-connector`**
4. Run the following installation command:

   `rpm –ivh gpfs.hadoop-connector-2.7.0-*.<platform-arch>.rpm`

5. **`mmhadoopctl connector attach --distribution BigInsights`**
6. Restart the IBM BigInsights IOP services over the current operating node.

Upgrading IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+) or 4.1.0.8 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+) or 4.1.0.8 releases, this section explains the steps required to upgrade the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

Also, ensure that there are no two versions of `hadoop-gpfs-*.jar` and `libgpfshadoop.so` under the Hadoop distribution, to avoid exceptions when running jobs over Hadoop.

1. **`mmhadoopctl connector stop`**
2. **`mmhadoopctl connector detach --distribution BigInsights`**
3. Run the following installation command:
   ```
   rpm –ivh --force gpfs.hadoop-connector-2.7.0-<your-new-build-number>.<platform-arch>.rpm
   ```
4. **`mmhadoopctl connector attach --distribution BigInsights`**
5. Restart the IBM BigInsights IOP services over the current operating node.

Upgrading IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+), 4.1.0.8, or 4.1.1+ releases

For users who are using IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+), 4.1.0.8, or 4.1.1+ releases, this section explains the steps required to upgrade the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

Also, ensure that there are no two versions of `hadoop-gpfs-*.jar` and `libgpfshadoop.so` under the Hadoop distribution, to avoid exceptions when running jobs over Hadoop.

1. **`mmhadoopctl connector stop`**
2. **`mmhadoopctl connector detach --distribution BigInsights`**
3. **`rpm -e gpfs.hadoop-connector`**
4. Run the following installation command:
   ```
   rpm –ivh gpfs.hadoop-connector-2.7.0-<your-new-build-number>.<platform-arch>.rpm
   ```
5. **`mmhadoopctl connector attach --distribution BigInsights`**
6. Restart the IBM BigInsights IOP services over the current operating node.

Deploy IBM Spectrum Scale using Ambari

IBM Spectrum Scale supports deploying IBM BigInsights IOP 4.1 together with IBM Spectrum Scale through Ambari 2.1. Integration of IBM BigInsights IOP and IBM Spectrum Scale leads to an easy deployment of their Hadoop cluster using the GUI mode.

You can download the `gpfs.ambari` rpm package from the IBM developerWorks GPFS Wiki . For more information, see the *DeployBigInsights4.1_SpectrumScale_with_Ambari 2.1* guide in the Wiki.

HDFS transparency

IBM Spectrum Scale HDFS transparency offers a set of interfaces that allows applications to use HDFS Client to access IBM Spectrum Scale through HDFS RPC requests.

All data transmission and metadata operations in HDFS are through RPC and processed by NameNode and DataNode services within HDFS. IBM Spectrum Scale HDFS transparency implementation integrates both NameNode and DataNode services and responds to the request as in HDFS. Advantages of HDFS transparency are as follows:

- HDFS Compliant APIs or shell-interface command.
- Application client isolation from storage. Application Client may access IBM Spectrum Scale without GPFS client installed.
- Improved security management by Kerberos authentication and encryption in RPC
- Simplified file system monitor by Hadoop Metrics2 integration

Supported IBM Spectrum Scale storage mode

Local Storage Mode

HDFS transparency allows big data applications to access IBM Spectrum Scale local storage mode-File Placement Optimizer (FPO) mode since gpfs.hdfs-protocol.2.7.0-0. HDFS transparency enables the support for shared storage mode such as SAN-based storage, ESS, etc.

In this mode, data blocks are stored in chunks in IBM Spectrum Scale, and replicated to protect against disk or node failure. DFS clients run over the storage node so it can leverage the data location for executing the task quickly. In such a storage mode, short-circuit read is recommended to improve the access efficiency.

Shared Storage Mode

HDFS transparency allows big data applications to access data stored in shared storage such SAN-based storage, ESS, etc.

In this mode, data is stored in SAN storage and it offers better storage efficiency than local storage. RAID and other technology can be used to protect hardware failure instead of taking replication.

When DFS Client requests to write blocks to IBM Spectrum Scale, HDFS transparency Name Node selects DataNode randomly for this request. Especially, when DFS Client is on one DataNode, the current node is selected for this request. When DFS Client requests to get block location of one existing block, NameNode selects 3 DataNodes randomly for this request.

HDFS transparency allows Hadoop application access data stored in both local spectrum scale file system and remote spectrum scale file system from multiple cluster.

Hadoop cluster planning

In an Hadoop cluster that runs the HDFS transparency, a node can take on the roles of DFS Client, NameNode, and DataNode. The Hadoop cluster might contain the whole IBM Spectrum Scale cluster or some of the nodes in the IBM Spectrum Scale cluster.

NameNode

You can specify one NameNode or multiple NameNodes to protect themselves against single node failure problems in a cluster. For more information, see "High availability configuration" on page 331. NameNode must be selected from the IBM Spectrum Scale cluster and must be a robust node to reduce the single-node failure. NameNode is configured as the hostname by `fs.defaultFS` in the `core-site.xml` file in Hadoop 2.4, 2.5, and 2.7 release.

DataNode

You can specify multiple DataNodes in a cluster. DataNodes must be within IBM Spectrum Scale cluster. DataNode are specified by the hostname in the `slaves` configuration file.

DFS Client

DFS Client can be placed within a IBM Spectrum Scale cluster or outside a IBM Spectrum Scale cluster. When placed within a IBM Spectrum Scale cluster, DFS Client can read data from IBM Spectrum Scale through RPC or short-circuit mode. Otherwise, DFS Client can access IBM Spectrum Scale only through RPC. You can specify the NameNode address in DFS Client configuration so that DFS Client can communicate with the appropriate NameNode service.

The purpose of cluster planning is to define the node roles: Hadoop node, HDFS transparency node, and GPFS node.

Node roles planning in FPO mode

In FPO mode, usually, all nodes are FPO nodes, Hadoop nodes, and HDFS transparency nodes. In this mode, Hadoop cluster must be larger than or equal to the HDFS transparency cluster. Hadoop cluster might be smaller than HDFS transparency cluster but this configuration is not typical and not recommended. Also, HDFS transparency cluster must be smaller than or equal to GPFS cluster because HDFS transparency needs to read/write data from the local mounted file system. Usually, in FPO mode, HDFS transparency cluster is equal to GPFS cluster.

Note: Some nodes in GPFS FPO cluster might be GPFS client without any disks in the file system.

Node roles planning in shared storage mode

If the IO stress is heavy, you might exclude the NSD servers from HDFS transparency cluster. Typically, in a shared storage mode, all nodes in Hadoop cluster might be GPFS client free, which means GPFS is not needed to be installed in these Hadoop nodes.

Note: All HDFS transparency nodes need to be installed with GPFS and mount the file systems

Installation and configuration of HDFS transparency

This section describes the installation and configuration of HDFS transparency.

Installation of HDFS transparency

IBM Spectrum Scale HDFS transparency must be installed on nodes that take roles of NameNode or DataNodes.

HDFS Transparency works with IBM Spectrum Scale 4.1 or later. For Linux distro version, see IBM Spectrum Scale FAQ in IBM Knowledge Center.

Use the following command to install the HDFS transparency:

```
# rpm -hiv gpfs.hdfs-protocol-2.7.0-0.x86_64.rpm
```

This package has the following dependencies:

- `Libacl`
- `Libattr`
- `openjdk 7.0+`

HDFS transparency is installed under the `/usr/lpp/mmfs/hadoop` folder. To list the contents of this directory, use the following command:

```
#ls /usr/lpp/mmfs/hadoop/
bin etc lib libexec license logs README run sbin share
```

You can add `bin` and `sbin` directory into the system path.

Configuration of HDFS transparency

For configuring, you must install Hadoop distribution under `$HADOOP_PREFIX` on each machine in the cluster. The configurations for GPFS HDFS transparency are located under `/usr/lpp/mmfs/hadoop/etc/hadoop` for any Hadoop distribution. Configurations for Hadoop distribution are located in different locations, for example, `/etc/hadoop/conf` for IBM BigInsights IOP.

The `core-site.xml` and `hdfs-site.xml` configuration files should be synced and kept identical for the IBM Spectrum Scale HDFS transparency and Hadoop distribution. For `slaves` and `log4j.properties`, the IBM Spectrum Scale HDFS transparency and Hadoop distribution can take different configuration.

OS tuning for all nodes:

ulimit tuning

For all nodes, `ulimit -n` and `ulimit -u` must be larger than or equal to 65536. Smaller value makes Hadoop java processes report unexpected exceptions.

In Red Hat, add the following lines at the end of `/etc/security/limits.conf`:

```
*       soft nofile 65536
*       hard nofile 65536

*       soft nproc 65536
*       hard nproc 65536
```

For other Linux systems, you might check the linux guide of your distro.

After the above change, you need to restart all services to make it effective.

Configure Hadoop Nodes:

For configuring Hadoop nodes, the following files need to be updated for open source Apache Hadoop:

- `core-site.xml`
- slaves

In this example, hostname of NameNode service is `hs22n44`:

In `$HADOOP_PREFIX/etc/hadoop/core-site.xml`, ensure that the configuration is

```
fs.defaultFS:
<property>
<name>fs.defaultFS</name>
<value>hdfs://hs22n44:9000</value>
</property>
```

Replace `hs22n44:9000` with the hostname of your NameNode service and preferred port number.

User can customize the other configurations like service port as HDFS. For more information, see http://hadoop.apache.org.

In `$HADOOP_PREFIX/etc/hadoop/slaves` file, ensure that all nodes are listed in the file. For example:

```
# cat $HADOOP_PREFIX/etc/hadoop/slaves
hs22n44
hs22n54
hs22n45
```

As for `hdfs-site.xml` and other detailed configuration in `core-site.xml`, follow the link http://hadoop.apache.org to configure Hadoop nodes. So far, the following must be configured to avoid unexpected exceptions from Hadoop:

```
<property>
<name>dfs.datanode.handler.count</name>
<value>40</value>
</property>

<property>
<name>dfs.datanode.handler.count</name>
<value>400</value>
</property>

<property>
<name>dfs.datanode.max.transfer.threads</name>
<value>8192</value>
</property>
```

After the configuration, sync them to all Hadoop nodes.

Note: If you take Hadoop distribution, like IBM BigInsights, you need to configure Hadoop components (e.g. HBase, Hive, oozie, etc) in the management GUI, e.g. Ambari for IBM BigInsights.

Configure HDFS transparency nodes:

Sync Hadoop configurations:

By default, HDFS transparency takes `core-site.xml` and `hdfs-site.xml` from the configuration of Hadoop distro, and `gpfs-site.xml` located under `/usr/lpp/mmfs/hadoop/etc/hadoop`.

On any one HDFS transparency node, assuming it is `hdfs_transparency_node1` here, if the node is also a Hadoop node, you might take the following command to sync `core-site.xml`, `hdfs-site.xml`, `slaves`, and `log4j.properties` from Hadoop configuration to HDFS transparency configuration directory (`/usr/lpp/mmfs/hadoop/etc/hadoop` by default):

```
hdfs_transparency_node1#/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf <hadoop-conf-dir>
```

If the node `hdfs_transparency_node1` is not a Hadoop node, you need to take scp to copy `<hadoop-conf-dir>/core-site.xml`, `<hadoop-conf-dir>/hdfs-site.xml`, `<hadoop-conf-dir>/log4j.properties` into `/usr/lpp/mmfs/hadoop/etc/hadoop/` on `hdfs_transparency_node1`.

Configure the storage mode:

Modify `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` on the node `hdfs_transparency_node1`:

```
<property>
<name>gpfs.storage.type</name>
<value>local</value>
</property>
```

The property *gpfs.storage.type* is used to specify storage mode. The storage mode can be local or shared. This is a required configuration and `gpfs-site.xml` must be synced to all HDFS transparency nodes after the above modification.

Update other configuration files:

Note: To configure Hadoop HDFS, Yarn, etc refer to the `hadoop.apache.org`. This section focuses on configurations related with HDFS.

Configurations for Apache Hadoop

Modify `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` on the node `hdfs_transparency_node1`:

```
<property>
<name>gpfs.mnt.dir</name>
<value>/gpfs_mount_point</value>
</property>

<property>
<name>gpfs.data.dir</name>
<value>data_dir</value>
</property>

<property>
<name>gpfs.supergroup</name>
<value>hadoop</value>
</property>

<property>
<name>gpfs.replica.enforced</name>
<value>dfs</value>
</property>
```

In `gpfs-site.xml`, all Hadoop data is stored under `/gpfs_mount_point/data_dir`. You can have two Hadoop clusters over the same file system and these clusters are isolated from each other. When Hadoop operates the file, one limitation is that if you have a link outside of `/gpfs_mount_point/data_dir` to inside of `/gpfs_mount_point/data_dir`, it reports an exception because Hadoop sees it as `/gpfs_mount_point/data_dir`. All files outside of `/gpfs_mount_point/data_dir` are not visible for Hadoop jobs.

`gpf.supergroup` must be configured according to your cluster. You need to add some Hadoop users, such as HDFS, yarn, hbase, hive, oozie, etc under the same group named Hadoop and configure `gpfs.supergroup` as Hadoop. You might specify two or more comma-separated groups as `gpfs.supergroup`, for example, `group1,group2,group3`.

Note: Users in `gpfs.supergroup` are super users and they can control all the data in `/gpfs_mount_point/data_dir`. This is similar to the user root in Linux.

`gpfs.replica.enforced` is used to control the replica rules. Hadoop controls the data replica by `dfs.replication`. When running Hadoop over IBM Spectrum Scale, IBM Spectrum Scale has its own replica rules. If you configure `gpfs.replica.enforced` as dfs , then `dfs.replication` is always effective unless you specify `dfs.replication` in the command options when submitting jobs. If you configure it into `gpfs.replica.enforced` as gpfs, then all data replication from GPFS self as `dfs.replication` is not effective. Usually, it is taken as dfs.

Usually, you must not change `core-site.xml` and `hdfs-site.xml` located under `/usr/lpp/mmfs/hadoop/etc/hadoop/`. These two files must be consistent as the files used by Hadoop nodes.

You need to modify `/usr/lpp/mmfs/hadoop/etc/hadoop/slaves` to add all HDFS transparency DataNode hostnames and one hostname per line, for example:

```
# cat /usr/lpp/mmfs/hadoop/etc/hadoop/slaves
hs22n44
hs22n54
hs22n45
```

You might check `/usr/lpp/mmfs/hadoop/etc/hadoop/log4j.properties` and modify it accordingly. This file might be different as the `log4j.properties` used by Hadoop nodes.

After you finish the configurations, use the following command to sync it to all IBM Spectrum Scale HDFS transparency nodes:

```
hdfs_transparency_node1#/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop
```

Configuration for IBM BigInsights IOP

In IBM BigInsights IOP 4.0/4.1, IOP and IBM Spectrum Scale are integrated manually. Therefore, it is easy to configure the IBM Spectrum Scale HDFS transparency. For IBM BigInsights IOP 4.1, if you deployed IOP 4.1 with IBM Spectrum Scale Ambari integration, you can email `scale@us.ibm.com` for more information. If you deployed IOP 4.1 without IBM Spectrum Scale Ambari integration, perform the following steps:

1. On the node `hdfs_transparency_node1`, run the following command to sync IBM BigInsights IOP configuration into IBM Spectrum Scale HDFS transparency configuration directory: **`/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf /etc/hadoop/conf/`**
2. On the node `hdfs_transparency_node1`, refer the Configurations for Apache Hadoop section to create the `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` file, update `/usr/lpp/mmfs/hadoop/etc/hadoop/slaves` and `/usr/lpp/mmfs/hadoop/etc/hadoop/log4j.properties`.
3. On the node `hdfs_transparency_node1`, run the **`/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf /usr/lpp/mmfs/ hadoop/etc/hadoop/`** command to sync the `gpfs-site.xml`, `core-site.xml`, `hdfs-site.xml`, slaves and `log4j.properties` to all the IBM Spectrum Scale HDFS transparency nodes.

How to update environment variables for HDFS transparency service:

This topic describes how to update environment variables for HDFS transparency service.

In some situations, the user needs to update this environment of Hadoop. For example, the user can change JVM option or update Hadoop environment variables like *HADOOP_LOG_DIR*. To edit this file:

1. On namenode, edit the `/usr/lpp/mmfs/Hadoop/etc/hadoop/hadoop-env.sh` and other files that are required.
2. Sync up the change to all the other nodes from `#/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop`

Start and stop the service

To start and stop HDFS transparency services, you need to be a root user. You also need to keep native HDFS service down because HDFS transparency provides the same services. If you keep both services up, it reports conflict in service network port number. You need to restart all other Hadoop services, such as Yarn, Hive, HBase, etc after you replace native HDFS with HDFS transparency.

To start the HDFS transparency service on the NameNode, use the following command:

```
/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector start
```

To stop the HDFS transparency service on the NameNode, use the following command:

```
/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector stop
```

Health check the service

Any user can conduct a health check of the Hadoop service.

To conduct a health check of the service, use the following command:

```
/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector getstate
```

High availability configuration

Manual HA switch configuration

The high availability (HA) implementation follows the HDFS High Availability feature using NFS. You can define a GPFS directory instead of an NFS directory to sync up information between two NameNodes.

In the following configuration example, the HDFS nameservice ID is mycluster and name node ID are nn1 and nn2. Configuration must be done in the core-site.xml file by defining fs.defaultFS with the nameservice ID.

```
<property>
<name>fs.defaultFS</name>
<value>hdfs://mycluster</value>
</property>
```

The following configuration must be done in the hdfs-site.xml file:

```
<property>
<!–define dfs.nameservices ID-->
<name>dfs.nameservices</name>
<value>mycluster</value>
</property>

<property>
<!–define name nodes ID for HA-->
<name>dfs.ha.namenodes.mycluster</name>
<value>nn1,nn2</value>
</property>

<property>
<!–Actual hostname and rpc address of name node ID-->
<name>dfs.namenode.rpc-address.mycluster.nn1</name>
<value>c8f2n06.gpfs.net:8020</value>
</property>

<property>
<!–Actual hostname and rpc address of name node ID-->
<name>dfs.namenode.rpc-address.mycluster.nn2</name>
<value>c8f2n07.gpfs.net:8020</value>
</property>

<property>
<!–Actual hostname and http address of name node ID-->
<name>dfs.namenode.http-address.mycluster.nn1</name>
<value>c8f2n06.gpfs.net:50070</value>
</property>

<property>
<!–Actual hostname and http address of name node ID-->
<name>dfs.namenode.http-address.mycluster.nn1</name>
<value>c8f2n07.gpfs.net:50070</value>
</property>

<property>
<!–Shared directory used for status sync up-->
<name>dfs.namenode.shared.edits.dir</name>
<value>/gpfs/bigfs/HA</value>
</property>
```

These configurations should be synced up into all nodes running HDFS transparency services. Then you can start the HDFS transparency service using the **mmhadoopctl** command.

After the service gets started, both NameNodes are in the standby mode, by default. You can activate one NameNode using the following command so that it responds to the client:

```
gpfs haadmin -transitionToActive --forceactive [name node ID]
```

For example, you can activate the nn1 NameNode by running the following command:

```
#/usr/lpp/mmfs/hadoop/bin/gpfs haadmin -transitionToActive –forceactive nn1
```

If the nn1 NameNode fails, you can activate another NameNode and relay the service by running the following command

```
#/usr/lpp/mmfs/hadoop/bin/gpfs haadmin -transitionToActive –forceactive nn2
```

Use the following command to get the status of the NameNode:

```
/usr/lpp/mmfs/hadoop/bin/gpfs haadmin -getServiceState [name node ID]
```

Automatic NameNode service HA

Automatic NameNode Service HA is supported in gpfs.hdfs-protocol 2.7.0-2 and later. The implementation of high availability (HA) is the same as NFS-based HA in native HDFS. The only difference is that except for the NFS shared directory in native HDFS, HA is not needed for HDFS transparency.

The prerequisite to configure automatic NameNode HA is to have zookeeper services running in the cluster.

Configuring Automatic NameNode Service HA:

If you take a Hadoop distro, such as IBM BigInsights IOP, the zookeeper service is deployed by default. However, if you select open-source Apache Hadoop, you must set up the zookeeper service by following the instruction on the zookeeper website.

Note: In the following configuration example, HDFS Transparency NameNode service ID is mycluster and NameNode IDs are nn1 and nn2. ZooKeeper server zk1.gpfs.net, zk2.gpfs.net and zk3.gpfs.net are configured to support automatic NameNode HA.

1. Define the name service ID in the core-site.xml that is used by your Hadoop distro.

   Note: If you are using IBM BigInsights IOP, you can change this configuration in Ambari GUI and restart the HDFS services to synchronize it with all the Hadoop nodes.

   ```
   <property>
   <name>fs.defaultFS</name>
   <value>hdfs://mycluster</value>
   </property>
   ```

2. Configure the hdfs-site.xml file used by your Hadoop distro:

   ```
   <property>
   <!--define dfs.nameservices ID-->
   <name>dfs.nameservices</name>
   <value>mycluster</value>
   </property>

   <property>
   <!--define name nodes ID for HA-->
   <name>dfs.ha.namenodes.mycluster</name>
   <value>nn1,nn2</value>
   </property>

   <property>
   <!--Actual hostname and rpc address of name node ID-->
   <name>dfs.namenode.rpc-address.mycluster.nn1</name>
   <value>c8f2n06.gpfs.net:8020</value>
   </property>

   <property>
   <!--Actual hostname and rpc address of name node ID-->
   <name>dfs.namenode.rpc-address.mycluster.nn2</name>
   <value>c8f2n07.gpfs.net:8020</value>
   </property>

   <property>
   <!--Actual hostname and http address of name node ID-->
   <name>dfs.namenode.http-address.mycluster.nn1</name>
   <value>c8f2n06.gpfs.net:50070</value>
   </property>

   ```

```
<property>
<!--Actual hostname and http address of name node ID-->
<name>dfs.namenode.http-address.mycluster.nn2</name>
<value>c8f2n07.gpfs.net:50070</value>
</property>

<property>
<!--Shared directory used for status sync up-->
<name>dfs.namenode.shared.edits.dir</name>
<value>/<gpfs.mnt.dir>/<gpfs.data.dir>/HA</value>
</property>

<property>
<name>dfs.ha.standby.checkpoints</name>
<value>false</value>
</property>

<property>
<name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>

<property>
<name>dfs.ha.fencing.methods</name>
<value>shell(/bin/true)</value>
</property>

<property>
<name>dfs.ha.automatic-failover.enabled</name>
<value>true</value>
</property>

<property>
<name>ha.zookeeper.quorum</name>
<value>zk1.gpfs.net:2181,zk2.gpfs.net:2181,zk3.gpfs.net:2181</value>
</property>
```

The configuration **dfs.namenode.shared.edits.dir** should be consistent with gpfs.mnt.dir and gpfs.data.dir defined in /usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml. You could create the directory /<gpfs.mnt.dir>/<gpfs.data.dir>/HA and make it owned by hdfs:hadoop before starting HDFS transparency services.

The **dfs.ha.standby.checkpoints** should be set as false. If not, you will see a log of exceptions in the standby NameNode logs, e.g.

```
ERROR ha.StandbyCheckpointer (StandbyCheckpointer.java:doWork(371)) - Exception in doCheckpoint.
```

HDFS transparency does not have fsImage and editLogs. Therefore, do not perform checkpoints from the standby NameNode service.

The configuration name dfs.client.failover.proxy.provider.mycluster must be changed according to the nameservice ID. In the above example, the nameservice ID is configured as *mycluster* in core-site.xml. Therefore, the configuration name is dfs.client.failover.proxy.provider.mycluster.

Note: If you enable Short Circuit Read, the value of this configuration must be *org.apache.hadoop.gpfs.server.namenode.ha.ConfiguredFailoverProxyProvider*.

3. To synchronize core-site.xml with hdfs-site.xml from your Hadoop distro to any one node that is running HDFS transparency services, see Sync Hadoop configurations.
4. On HDFS_Transparency_node1, modify the /usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml:

```
<property>
<name>dfs.client.failover.proxy.provider.mycluster</name>
<value>org.apache.hadoop.gpfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

The WebHDFS service functions properly when NameNode HA is enabled.

Note: On HDFS transparency nodes, the above configuration value in hdfs-site.xml is different as that in Step2.

5. On HDFS_Transparency_node1, run the following command as the root user to synchronize HDFS Transparency configuration with all HDFS transparency nodes:
   ```
   mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop
   ```
6. Start the HDFS Transparency service by running the **mmhadoopctl** command:
   ```
   mmhadoopctl connector start
   ```
7. Start the zkfc daemon:
   ```
   /usr/lpp/mmfs/hadoop/sbin/hadoop-daemon.sh start zkfc -formatZK
   ```
   You can remove the formatZK option. Run jps on the nn1 and nn2 name nodes to check if the DFSZKFailoverController process has been started.

   Note: If the -formatZK option is not added, the system displays the following exception: `FATAL org.apache.hadoop.ha.ZKFailoverController: Unable to start failover controller. Parent znode does not exist`
8. Run the following command to check that all NameNode services and DataNode services are functioning:
   ```
   mmhadoopctl connector getstate
   ```
9. Run the following command to check the state of NameNode services:
   ```
   /usr/lpp/mmfs/hadoop/bin/gpfs haadmin -getServiceState [name node ID]
   ```

   Note: When HA is enabled for HDFS transparency, the following exception might be logged: `Get corrupt file blocks returned error: Operation category READ is not supported in state standby`. These are unfixed HDFS issues: HDFS-3447 and HDFS-8910.

Short-circuit read configuration

In HDFS, read requests go through the DataNode. When the client asks the DataNode to read a file, the DataNode reads that file off the disk and sends the data to the client over a TCP socket. The short-circuit read obtains the file descriptor from the DataNode, allowing the client to read the file directly.

This is possible only in cases where the client is co-located with the data and used in the FPO mode. Short-circuit reads provide a substantial performance boost to many applications.

Note: Short-circuit local reads can only be enabled on Hadoop 2.7.0. For more information on how to enable short-circuit reads on other Hadoop versions, contact `scale@us.ibm.com`.

Configuring short-circuit local read

To configure short-circuit local reads, you need to enable `libhadoop.so` and use the DFS Client shipped by the IBM Spectrum Scale HDFS transparency, the package name is `gpfs.hdfs-protocol`. You cannot use standard HDFS DFS Client to enable the short-circuit mode over the HDFS transparency.

To enable `libhadoop.so`, compile the native library on the target machine or use the library shipped by IBM Spectrum Scale HDFS transparency. To compile the native library on the specific machine, do the following steps:

1. Download Hadoop source code from Hadoop community
2. Build by `mvn`: **`$ mvn package -Pdist,native -DskipTests -Dtar`**
3. Copy `hadoop-dist/target/hadoop-2.7.1/lib/native/libhadoop.so.*` to `$HADOOP_PREFIX/lib/native/`

   Or, to use the `libhadoop.so` delivered by the HDFS transparency, copy `/usr/lpp/mmfs/hadoop/lib/native/libhadoop.so` to `$HADOOP_PREFIX /lib/native/libhadoop.so`

   The shipped `libhadoop.so` is built on `x86_64`, `ppc64`, or `ppc64le` respectively.

   Note: This step must be done over all nodes running the Hadoop tasks.

Enabling DFS Client:
To enable DFS Client, shipped along with the HDFS transparency, on each node that accesses IBM Spectrum Scale in the short-circuit mode:

1. Back up **hadoop-hdfs-2.7.0.jar** using `$ mv $HADOOP_PREFIX/share/hadoop/hdfs/hadoop-hdfs-2.7.0.jar $HADOOP_PREFIX/share/hadoop/hdfs/hadoop-hdfs-2.7.0.jar.backup`
2. Link **hadoop-gpfs-2.7.0.jar** to classpath `$ln -s /usr/lpp/mmfs/hadoop/share/hadoop/hdfs/hadoop-gpfs-2.7.0.jar $HADOOP_PREFIX/share/hadoop/hdfs/hadoop-gpfs-2.7.0.jar`
3. Update the `core-site.xml` file with the following information:

```
<property>
  <name>fs.hdfs.impl</name>
  <value>org.apache.hadoop.gpfs.DistributedFileSystem</value>
</property>
```

Short-circuit reads make use of a UNIX domain socket. This is a special path in the file system that allows the client and the DataNodes to communicate. You need to set a path to this socket. The DataNode needs to be able to create this path. However, it must not be possible for any user except the HDFS user or root to create this path. Therefore, paths under `/var/run` or `/var/lib` folders are often used.

The client and the DataNode exchange information through a shared memory segment on the `/dev/shm` path. Short-circuit local reads need to be configured on both the DataNode and the client. Here is an example configuration.

```
<configuration>
<property>
<name>dfs.client.read.shortcircuit</name>
<value>true</value>
</property>
<property>
<name>dfs.domain.socket.path</name>
<value>/var/lib/hadoop-hdfs/dn_socket</value>
</property>
</configuration>
```

Sync up all these changes in the entire cluster and if needed, restart the service.

Note: The `/var/lib/hadoop-hdfs` and `dfs.domain.socket.path` must be created manually by the root user before running the short-circuit read. The `/var/lib/hadoop-hdfs` must be owned by the root user. If not, the DataNode service fails when starting up.

```
#mkdir -p  /var/lib/hadoop-hdfs
#chown root:root /var/lib/hadoop-hdfs
#touch   /var/lib/hadoop-hdfs/${dfs.dome.socket.path}
#chmod 666 /var/lib/hadoop-hdfs/${dfs.dome.socket.path}
```

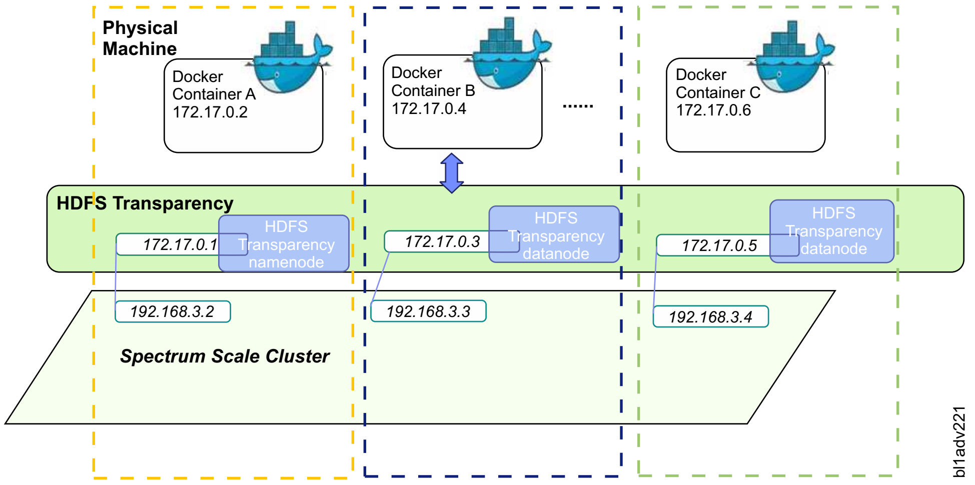
The permission control in short-circuit reads is similar to the common user access in HDFS. If you have the permission to read the file, then you can access it through short-circuit read.

| ## Docker support

| HDFS transparency supports running the Hadoop Map/Reduce workload inside the virtual machine
| container, Docker.

| See the Docker website for Docker technology.

| With HDFS transparency, you can run Hadoop Map/Reduce jobs in Docker and take IBM Spectrum Scale
| as the uniform data storage layer over the physical machines.


Physical Machine
Docker Container A 172.17.0.2
Docker Container B 172.17.0.4
......
Docker Container C 172.17.0.6
HDFS Transparency
172.17.0.1
HDFS Transparency namenode
172.17.0.3
HDFS Transparency datanode
172.17.0.5
HDFS Transparency datanode
192.168.3.2
192.168.3.3
192.168.3.4
Spectrum Scale Cluster
bl1adv221

You can configure different Docker instances from different physical machines as one Hadoop cluster and run Map/Reduce jobs on the virtual Hadoop clusters. All Hadoop data is stored in the IBM Spectrum Scale file system over the physical machines. The `172.17.0.x` IP address over each physical machine is a network bridge adapter used for network communication among Docker instances from different physical machines. HDFS transparency services must be configured to monitor the network bridge and process the requests from Docker instances. After receiving the requests from Hadoop jobs running in Docker instances, HDFS transparency handles the I/O requests for the mounted IBM Spectrum Scale file system on the node.

Configuring the Docker instance and HDFS transparency

1. Docker (version 1.9+) requires Redhat7+. Modify the Redhat Yum Repos to upgrade the selinux-policy and device-mapper-libs by running the following commands:
   - **`yum upgrade selinux-policy`**
   - **`yum upgrade device-mapper-libs`**
2. To install Docker engine (version 1.9+), See link.
3. Configure the network bridge adapter on physical machines. There can be only one network bridge adapter on one machine.

   Note: These configurations must be changed under `/etc/sysconfig/network-scripts/`:

```
[root@c3m3n04 network-scripts]# cat ifcfg-br0
DEVICE=br0
TYPE=Bridge
BOOTRPOTO=static
IPADDR=172.17.0.1
NETMASK=255.255.255.0
ONBOOT=yes

[root@c3m3n04 network-scripts]# cat ifcfg-enp11s0f0
# Generated by dracut initrd
DEVICE="enp11s0f0"
ONBOOT=yes
NETBOOT=yes
UUID="ca481ab0-4cdf-482e-b5d3-82be13a7621c"
IPV6INIT=yes
BOOTPROTO=static
HWADDR="e4:1f:13:be:5c:28"
TYPE=Ethernet
NAME="enp11s0f0"
IPADDR=192.168.3.2
BROADCAST=192.168.255.255
NETMASK=255.255.255.0
```

Note: You must modify the IPADDR, BROADCAST, and NETMASK according to your network configuration.

In this example, the br0 bridge adapter is bundled with the enp11s0f0 physical adapter. You must modify the code in the example for all the physical machines on which the Docker instances must be run.

4. Modify the Docker service script and start the Docker engine daemons on each node:

```
vim /usr/lib/systemd/system/docker.service
ExecStart=/usr/bin/docker daemon -b br0 -H fd://

service docker stop
service docker start
```

5. Configure the network route table on each machine:

```
route add -net 172.17.1.0/24 gw <replace-physical-node-ip-here> dev enp11s0f0
```

where <replace-physical-node-ip-here> is the IP address of your machine.

6. The IP addresses of the nodes must be different so that the Docker instances from one physical node can access the Docker instances in another physical node. Check if you can connect to the br0 IP address from another node.

7. Configure HDFS transparency and start the HDFS transparency services. Modify `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xlm` and `/usr/lpp/mmfs/hadoop/etc/hadoop/slaves`. You must select the IP address from Docker network bridge adapter. Pull the Hadoop Docker image on each node:

```
docker pull sequenceiq/hadoop-docker:2.7.0
```

Note: We have selected the Hadoop Docker image from sequenceiq.

8. Start all Docker instances on each node by running the following command:

```
#docker run -h <this-docker-instance-hostname> -it sequenceiq/hadoop-docker:2.7.0
/etc/bootstrap.sh -bash
```

You can start multiple Docker instances over the same physical node. This command starts a Docker instance with the hostname <this-docker-instance-hostname>.

9. For each Docker instance, change the `/etc/hosts` to map the Docker instance IP addresses to the hostname:

```
#vi /etc/hosts
172.17.0.2 node1docker1.gpfs.net node1docker1
172.17.0.4 node2docker1.gpfs.net node2docker1
172.17.0.6 node3docker1.gpfs.net node3docker1
```

Note: This must be done on the console of each Docker instance. You must add all Docker instances here if you want to set them up as one Hadoop cluster.

After a Docker instance is stopped, all changes are lost and you will have to make this change again after a new Docker instance has been started.

10. Select a Docker instance and start the Yarn ResourceManager on it:

```
#cd /usr/local/hadoop-2.7.0/sbin ./start-yarn.sh
```

You cannot run two ResourceManagers in the same Hadoop cluster. Therefore, you run this ResourceManager in the selected Docker instance.

11. Start Yarn NodeManager on other Docker instances by running the following command:

```
#/usr/local/hadoop-2.7.0/sbin/yarn-daemon.sh --config /usr/local/hadoop/etc/hadoop/
start nodemanager
```

12. Run `hadoop dfs -ls /` to check if you can run Map/Reduce jobs in Docker now. To stop the Yarn services running in Docker, perform the following steps:

```
->on Yarn ResourceManager Docker instance:
cd /usr/local/hadoop-2.7.0/sbin ./stop-yarn.sh
->on Yarn NodeManager Docker instances:
 /usr/local/hadoop-2.7.0/sbin/yarn-daemon.sh --config /usr/local/hadoop/etc/hadoop/
stop nodemanager
```

Note: While selecting HDFS transparency, the data locality is not supported for the Map/Reduce jobs running in Docker.

The HDFS transparency federation

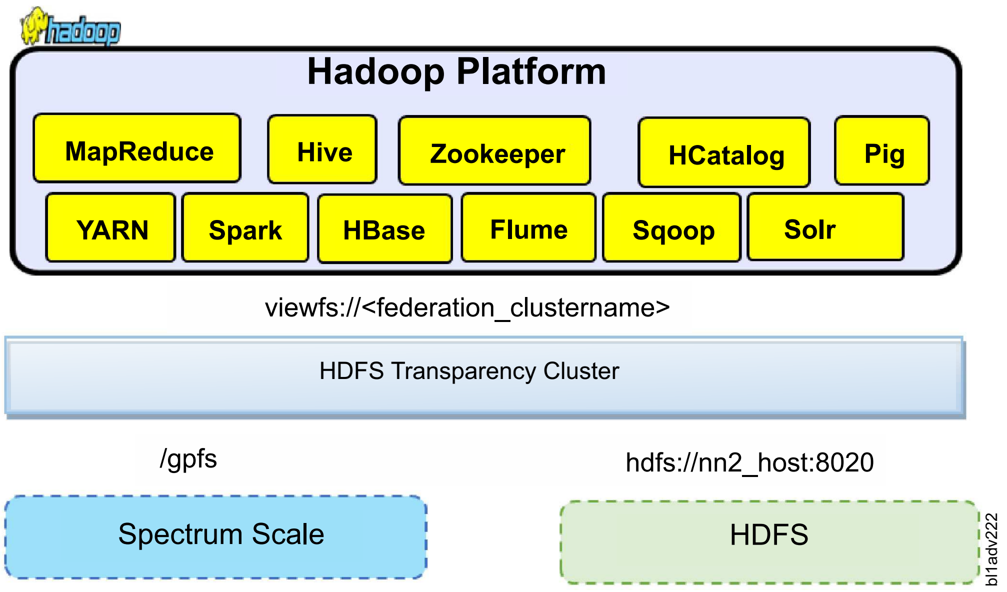
Federation has been introduced in HDFS to solve the HDFS NameNode scaling problem. This topic provides an overview of the HDFS Federation feature and the configuration and management of the federated cluster.

In HDFS transparency, federation is used to make the IBM Spectrum Scale file system coexist with the HDFS file system. For example, the Hadoop applications can get input from the native HDFS, analyze, and send the output to IBM Spectrum Scale. This feature is available in HDFS transparency 2.7.0-2 (`gpfs.hdfs-protocol-2.7.0-2`) and later. Also, the HDFS transparency federation can make two or more IBM Spectrum Scale file systems as one uniform file system for Hadoop applications. This is possible even if the file systems are from the same cluster as well as from different clusters. In a typical scenario, if you want to read data from an existing IBM Spectrum Scale file system, and analyze and send the analysis results to the new IBM Spectrum Scale file system.

Federating IBM Spectrum Scale and HDFS

This topic lists the steps for federating IBM Spectrum Scale and HDFS.

Ensure that you have configured the HDFS Transparency cluster.



1. On one HDFS transparency cluster node, shut down the HDFS transparency cluster daemon by running the following command:

   `mmhadoopctl connector stop`

2. On *nn1_host*, add the following configuration settings in `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml`:

   ```
   <configuration>
   <property>
      <name>fs.defaultFS</name>
   ```

```
   <value>viewfs://<federation_clustername></value>
   <description>The name of the federation file system</description>
</property>


<property>
   <name>fs.viewfs.mounttable.<federation_clustername>.link./<mount_dir></name>
   <value>hdfs://nn1_host:8020/<mount_dir></value>
   <description>The name of the Spectrum Scale file system</description>
</property>


<property>
   <name>fs.viewfs.mounttable.<federation_clustername>.link./<mount_dir></name>
   <value>hdfs://nn2_host:8020/<mount_dir></value>
   <description>The name of the hdfs file system</description>
</property>
</configuration>
```

Note: Change `<federation_clustername>` and `<mount_dir>` according to your cluster configuration: *nn1_host* and *nn2_host*. Here, *nn1_host* is the HDFS Transparency NameNode and *nn2_host* is the native HDFS NameNode.

3. On *nn1_host*, add the following configuration settings in `/usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml`:

```
<configuration>
<property>
   <name>dfs.nameservices</name>
   <value>nn1,nn2</value>
</property>

<property>
   <name>dfs.namenode.rpc-address.nn1</name>
   <value>nn1-host:8020</value>
</property>

<property>
   <name>dfs.namenode.rpc-address.nn2</name>
   <value>nn2-host:8020</value>
</property>

<property>
   <name> dfs.namenode.http-address.nn1</name>
   <value>nn1-host:50070</value>
</property>

<property>
   <name>dfs.namenode.http-address.nn2</name>
   <value>nn2-host:50070</value>
</property>

</configuration>
```

4. On *nn1_host*, synchronize the configuration changes with the other HDFS transparency nodes by running the following command:

   `mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop/`

5. On *nn1_host*, start all the HDFS transparency cluster nodes by running the following command:

   `mmhadoopctl connector start`

6. Stop the Hadoop applications and the native HDFS services on the native HDFS cluster. The detailed command is dependent on the Hadoop distro. For example, for IBM BigInsights IOP, stop all the services on Ambari GUI.

7. Perform Step 2 and Step 3 on the node *nn2-host* with correct path of `core-site.xml` and `hdfs-site.xml` according to the Hadoop distro.

If you take open source Apache Hadoop, the location of `core-site.xml` and `hdfs-site.xml` is `$YOUR_HADOOP_PREFIX/etc/hadoop/`. The $YOUR_HADOOP_PREFIX is the location of the Hadoop package. If you select IBM BigInsights IOP, you must update the configurations of `hdfs-site.xml` and `core-site.xml` on Ambari GUI.

8. Synchronize the updated configurations from the node *nn2-host* with all other native HDFS nodes and start the native HDFS services.

   If you select open source Apache Hadoop, you need to take scp to synchronize `core-site.xml` and `hdfs-site.xml` from the host *nn2-host* into all other native HDFS nodes. Then, start the native HDFS service by running `$YOUR_HADOOP_PREFIX/sbin/start-dfs.sh`.

   If you take IBM BigInsights IOP, you need to restart the native HDFS on Ambari GUI. Ambari then synchronizes all configurations into all Hadoop nodes. If native HDFS services are up, you must shutdown native HDFS services again to ensure that all services on native HDFS are stopped. You could check the configurations under `/etc/hadoop/conf` to ensure that all changes have been synchronized into all nodes.

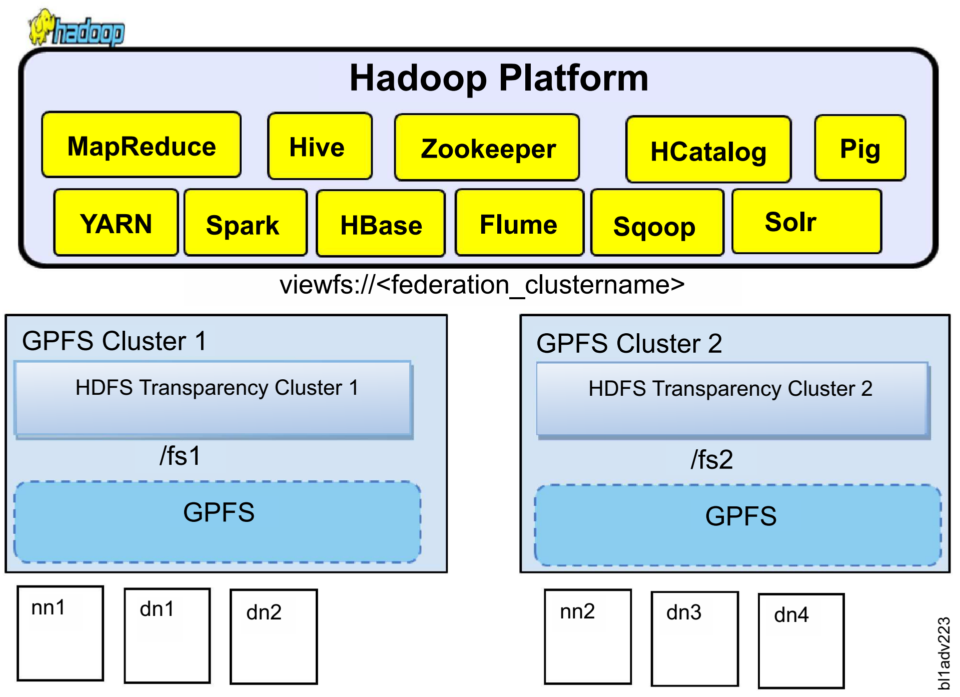
9. Update `core-site.xml` and `hdfs-site.xml` for the Hadoop clients on which the hadoop applications will run over federation. If you select open source Apache Hadoop, the location of `core-site.xml` and `hdfs-site.xml` is `$YOUR_HADOOP_PREFIX/etc/hadoop/`. The $YOUR_HADOOP_PREFIX is the location of the Hadoop package. If you take other Hadoop distro, see “Known limitations” on page 343.

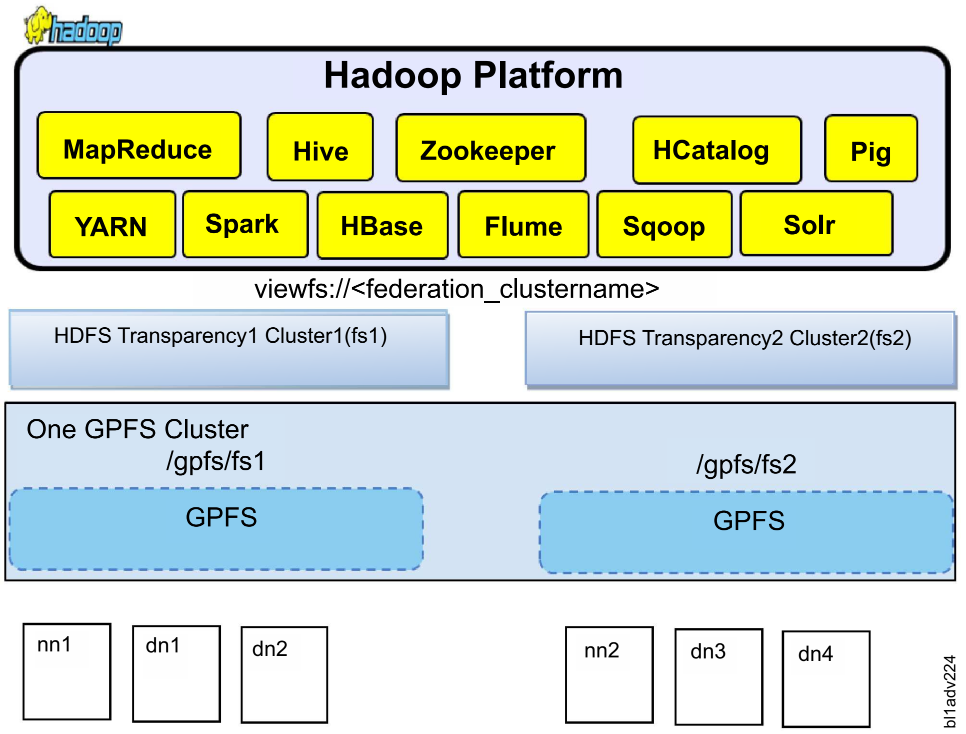
10. To ensure that the federated fs is functioning correctly, run the **`hadoop fs –ls /`** command.

Spectrum Scale file systems federation

You can federate two IBM Spectrum Scale file systems from different clusters or from the same cluster.

Irrespective of the mode that you select, configure one HDFS transparency cluster for each IBM Spectrum Scale file system, and then federate the two HDFS transparency clusters together.





To federate two file systems from the same cluster, select nodes that can provide HDFS transparency services for the first file system and the second file system separately.

Configuring the federation:

This topic describes the steps to configure the federation

Before configuring the federation, configure HDFS transparency cluster 1 and HDFS transparency cluster 2 for each file system.

1. To stop the services, run the **mmhadoopctl** connector stop on both HDFS transparency clusters.
2. On the *nn1* host, add the following configuration settings in /usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml:

```
<configuration>
<property>
   <name>fs.defaultFS</name>
   <value>viewfs://<federation_clustername></value>
   <description>The name of the federation file system</description>
</property>


<property>
   <name>fs.viewfs.mounttable.<federation_clustername>.link./<mount_dir></name>
   <value>hdfs://nn1_host:8020/<mount_dir></value>
   <description>The name of the gpfs file system</description>
</property>


<property>
   <name>fs.viewfs.mounttable.<federation_clustername>.link./<mount_dir></name>
   <value>hdfs://nn2_host:8020/<mount_dir></value>
   <description>The name of the hdfs file system</description>
</property>
</configuration>
```

Note: Change <federation_clustername> and <mount_dir> according to your cluster. Change *nn1_host* and *nn2_host* accordingly.

3. On *nn1_host*, add the following configuration settings in hdfs-site.xml.

```
<configuration>
<property>
   <name>dfs.nameservices</name>
   <value>nn1,nn2</value>
</property>

<property>
   <name>dfs.namenode.rpc-address.nn1</name>
   <value>nn1-host:8020</value>
</property>

<property>
   <name>dfs.namenode.rpc-address.nn2</name>
   <value>nn2-host:8020</value>
</property>

<property>
   <name> dfs.namenode.http-address.nn1</name>
   <value>nn1-host:50070</value>
</property>

<property>
   <name>dfs.namenode.http-address.nn2</name>
   <value>nn2-host:50070</value>
</property>
</configuration>
```

4. On *nn1_host*, synchronize the configuration change to another HDFS transparency cluster node:

   mmhadoopctl connector syncconf /usr/lpp/mmfs/hadoop/etc/hadoop/

5. On *nn2_host*, perform Step 1 through Step 4.
6. On *nn1_host*, start the HDFS transparency cluster by running the following command:

   mmhadoopctl connector start

7. On *nn2_host*, start the other HDFS transparency cluster by running the following command:

   mmhadoopctl connector start

8. Update core-site.xml and hdfs-site.xml for the Hadoop clients on which the Hadoop applications run over federation.

   If you take open source Apache Hadoop, the location of core-site.xml and hdfs-site.xml is $YOUR_HADOOP_PREFIX/etc/hadoop/. The $YOUR_HADOOP_PREFIX is the location of the Hadoop package. If you take another Hadoop distro, see "Known limitations."

9. Restart the Hadoop applications on both clusters.

   Note: If you select HDFS Transparency, you must always keep the native HDFS non-functional.

10. To ensure that the federated fs is functioning correctly, run the **hadoop fs –ls /** command.

Known limitations

- All the changes in /usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml and /usr/lpp/mmfs/hadoop/etc/hadoop/hdfs-site.xml must be updated in the configuration file that is used by the Hadoop distro. However, Hadoop distro occasionally manages the configuration, and the management interface might not support the key used for federation. IBM BigInsights IOP takes Ambari and Ambari GUI does not support some property names (see Ambari-15455).

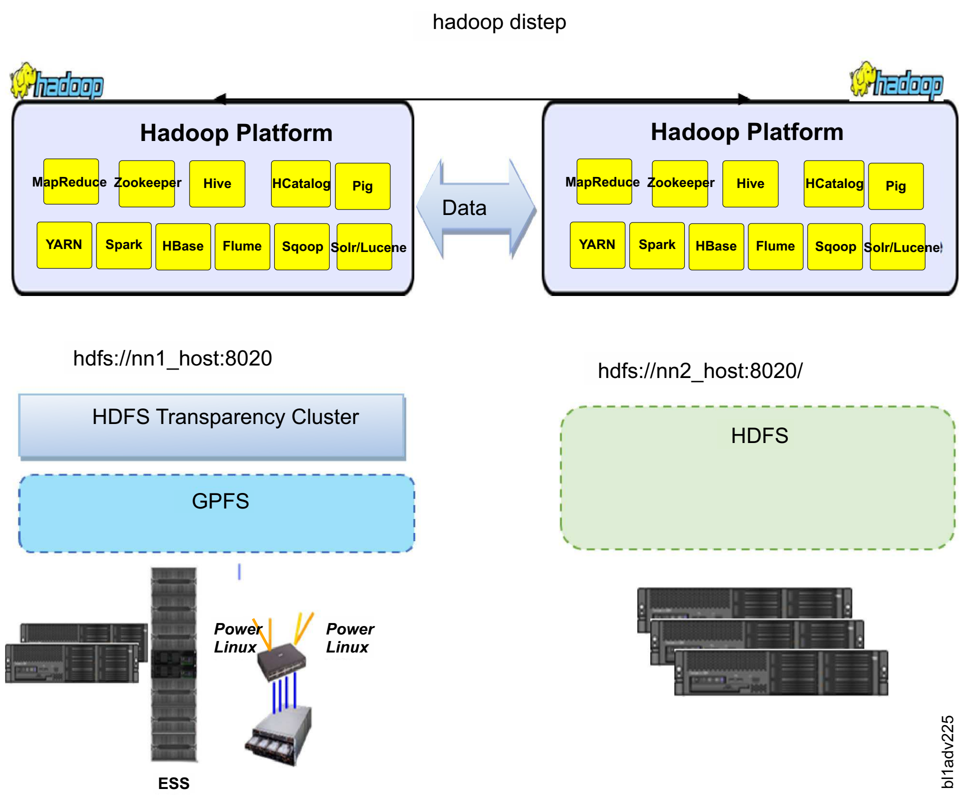
  If you want to set up the federation for IBM BigInsights IOP, send an email to scale@us.ibm.com.

- The native HDFS and HDFS transparency cannot be run over the same node because of the network port number conflict.

- Before being federated to GPFS, native HDFS must be in the running state. Otherwise, HDFS reports an exception.
- Start and stop the native HDFS cluster or the HDFS Transparency cluster separately if you want to maintain them.

Hadoop distcp support

The **hadoop distcp** command is used for data migration from HDFS to the IBM Spectrum Scale file system and between two IBM Spectrum Scale file systems.



There are no additional configuration changes. The **hadoop distcp** command is supported in HDFS transparency 2.7.0-2 (gpfs.hdfs-protocol-2.7.0-2) and later.

```
hadoop distcp hdfs://nn1_host:8020/source/dir hdfs://nn2_host.:8020/target/dir
```

Known Issues and Workaround

Issue 1: Permission is denied when the `hadoop distcp` command is run with the root credentials.

The super user root in Linux is not the super user for Hadoop. If you do not add the super user account to gpfs.supergroup, the system displays the following error message:

```
org.apache.hadoop.security.AccessControlException: Permission denied: user=root, access=WRITE,
inode="/user/root/.staging":hdfs:hdfs:drwxr-xr-x
```

at
`org.apache.hadoop.hdfs.server.namenode.FSPermissionChecker.check(FSPermissionChecker.java:319).`

Workaround

Add the super user account to gpfs.supergroup in gpfs-site.xml to configure the root as the super user or run the related **hadoop distcp** command with the super user credentials.

Issue 2: Access time exception while copying files from IBM Spectrum Scale to HDFS with the -p option

```
[hdfs@c8f2n03 conf]$ hadoop distcp -overwrite -p hdfs://c16f1n03.gpfs.net:8020/testc16f1n03/
hdfs://c8f2n03.gpfs.net:8020/testc8f2n03
```

Error: `org.apache.hadoop.ipc.RemoteException(java.io.IOException)`: Access time for HDFS is not configured. Set the **dfs.namenode.accesstime.precision** configuration parameter at `org.apache.hadoop.hdfs.server.namenode.FSDirAttrOp.setTimes(FSDirAttrOp.java:101)`

Workaround

Change the `dfs.namenode.accesstime.precision` value from 0 to a value such as 3600000 (1 hour) in `hdfs-site.xml` for the HDFS cluster.

Issue 3: The distcp command fails when the src director is root.

```
[hdfs@c16f1n03 root]$ hadoop distcp hdfs://c16f1n03.gpfs.net:8020/ hdfs://c8f2n03.gpfs.net:8020/test5
16/03/03 22:27:34 ERROR tools.DistCp: Exception encountered
java.lang.NullPointerException
```

at `org.apache.hadoop.tools.util.DistCpUtils.getRelativePath(DistCpUtils.java:144)`

at `org.apache.hadoop.tools.SimpleCopyListing.writeToFileListing(SimpleCopyListing.java:353)`

Workaround

Specify at least one directory or file at the source directory.

Issue 4: The distcp command throws NullPointerException when the target directory is root in the federation configuration but the job is completed

The **hadoop distcp** command throws NullPointerException when the target directory in the federation configuration is root, and the job is completed. For more details, see https://issues.apache.org/jira/browse/HADOOP-11724.

Automatic Configuration Refresh

The Automatic configuration refresh feature is supported in gpfs.hdfs-protocol 2.7.0-2 and later.

After making configuration changes in `/usr/lpp/mmfs/hadoop/etc/hadoop` or in the IBM Spectrum Scale file system, such as maximum number of replica and NSD server, run the following command to refresh HDFS transparency without restarting the HDFS transparency services:

```
/usr/lpp/mmfs/hadoop/bin/gpfs dfsadmin -refresh <namenode_hostname>:<port> refreshGPFSConfig
```

Run the command on an HDFS transparency node and change *<namenode_hostname>:<port>* according to the HDFS transparency configuration. For example, if *fs.defaultFS* is `hdfs://c8f2n03.gpfs.net:8020` in `/usr/lpp/mmfs/hadoop/etc/hadoop/core-site.xml`, replace *<namenode_hostname>* with `c8f2n03.gpfs.net` and *<port>* with `8020`. HDFS transparency synchronizes the configuration changes with the HDFS transparency services running on the HDFS transparency nodes.

Application interaction with HDFS transparency

The Hadoop application interacts with the HDFS transparency similar to their interactions with HDFS. They can access HDFS using Hadoop file system APIs and DistributedFileSystem APIs.

The application might have its own cluster that is larger than HDFS transparency cluster. However, all the nodes within the application cluster must be able to connect to all nodes in HDFS transparency cluster by RPC.

Yarn can define the nodes in cluster by using the slave files. However, HDFS transparency can use a set of configuration files that are different from yarn. In that case, slave files in HDFS transparency can be different from the one in the yarn.

Application interface of HDFS transparency

In HDFS transparency, applications can use the APIs defined in org.apache.hadoop.fs.FileSystem class and org.apache.hadoop.fs.AbstractFileSystem class to access the file system.

Command line for HDFS transparency

You can use the HDFS shell command line in the HDFS transparency.

You can access commands from the HDFS command shell:

```
$HADOOP_PREFIX/bin/hdfs
Usage: hdfs  [--config confdir] [--loglevel loglevel] COMMAND
       where COMMAND is one of:
  dfs                  run a filesystem command on the file systems supported in Hadoop.
  classpath            prints the classpath
  namenode             run the DFS namenode
  datanode             run a DFS datanode
  dfsadmin             run a DFS admin client
  haadmin              run a DFS HA admin client
  version              print the version
```

Most commands print help when invoked without parameters.

Security

HDFS transparency supports full Kerberos and it is verified over IBM BigInsights IOP 4.1. Refer to the IBM Spectrum Scale HDFS transparency Security Guide .

Hadoop Data Access Audit

HDFS Transparency is certificated with IBM Security Guardium® DAM (Database Activity Monitoring) to monitor the Hadoop Data Access over IBM Spectrum Scale. For more information, see Big data security and auditing with IBM InfoSphere Guardium.

Removing and upgrading HDFS transparency cluster

Before upgrading the HDFS transparency, you need to remove the older IBM Spectrum Scale Hadoop connector.

Removing IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.4, 4.1.0.5, 4.1.0.6, or 4.1.0.7 releases, this section explains the steps required to remove the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

1. Remove any links or copies of the `hadoop-gpfs-2.4.jar` file from your Hadoop distribution directory. Also, remove any links or copies of the `libgpfshadoop.64.so` file from your Hadoop distribution directory.

Note: For IBM BigInsights IOP 4.0, the distribution directory is `/usr/iop/4.0.0.0`.

2. Stop the current connector daemon:

   ```
   # ps -elf | grep gpfs-connector-daemon
   # kill -9 <pid-of-connector-daemon>
   ```

3. Run the following commands, to remove callbacks from IBM Spectrum Scale:

   ```
   cd /usr/lpp/mmfs/fpo/hadoop-2.4/install_script
   ./gpfs-callbacks.sh --delete
   ```

   Run the **`mmlscallbacks all`** command to check whether connector-related callbacks, such as `callback ID` `start-connector-daemon` and `stop-connector-daemon`, are removed. The IBM Spectrum Scale Hadoop connector callbacks are cluster-wide and this step is required to be done over any one of nodes.

4. Remove the following files:

   ```
   rm -f /var/mmfs/etc/gpfs-callbacks.sh
   rm -f /var/mmfs/etc/gpfs-callback_start_connector_daemon.sh
   rm -f /var/mmfs/etc/gpfs-callback_stop_connector_daemon.sh
   rm -f /var/mmfs/etc/gpfs-connector-daemon
   ```

5. Remove the IBM Spectrum™ Scale-specific configuration from your Hadoop `core-site.xml` file. Modify the **`fs.defaultFS`** into an HDFS schema format after removing the following configurations:

   ```
   fs.AbstractFileSystem.gpfs.impl, fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl,
   gpfs.mount.dir, gpfs.supergroup
   ```

Install and setup the HDFS transparency, see the "Upgrading HDFS transparency cluster" on page 348 for more information.

Removing IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.7 (efix3) or 4.1.0.8 releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 over IBM Spectrum Scale 4.1.0.7 (efix3) or 4.1.0.8 releases, this section explains the steps required to remove the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

1. **`mmhadoopctl connector stop`**
2. **`mmhadoopctl connector detach --distribution BigInsights`**
3. **`rpm -e gpfs.hadoop-2-connector`**
4. Remove the IBM Spectrum Scale-specific configuration from your Hadoop `core-site.xml` file. Modify the **`fs.defaultFS`** into an HDFS schema format by removing the following configurations:

   ```
   fs.AbstractFileSystem.gpfs.impl, fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl,
   gpfs.mount.dir, gpfs.supergroup
   ```

Install and setup the HDFS transparency, see the "Upgrading HDFS transparency cluster" on page 348 for more information.

Removing IBM Spectrum Scale Hadoop connector 2.4 or 2.5 over IBM Spectrum Scale 4.1.1+ releases

For users who are using IBM Spectrum Scale Hadoop connector 2.4 or 2.5 in IBM Spectrum Scale 4.1.1 and later, this section explains the steps required to remove the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

1. **`mmhadoopctl connector stop`**
2. **`mmhadoopctl connector detach --distribution BigInsights`**

3. **rpm -e gpfs.hadoop-2-connector**
4. Remove the IBM Spectrum Scale-specific configuration from your Hadoop `core-site.xml` file. Modify the **fs.defaultFS** into an HDFS schema format by removing the following configurations:

   ```
   fs.AbstractFileSystem.gpfs.impl, fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl,
   gpfs.mount.dir, gpfs.supergroup
   ```

Install and setup the HDFS transparency, see the "Upgrading HDFS transparency cluster" for more information.

Removing IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+), 4.1.0.8, or 4.1.1+ releases

For users who are using IBM Spectrum Scale Hadoop connector 2.7 (earlier release) over IBM Spectrum Scale 4.1.0.7 (efix3+), 4.1.0.8, or 4.1.1+ releases, this section explains the steps required to remove the old connector over each node in the cluster.

If you are using Hadoop 1.x, IBM Spectrum Scale Hadoop Connector 2.7 does not support Hadoop 1.x and you need to upgrade your Hadoop version first.

1. **mmhadoopctl connector stop**
2. **mmhadoopctl connector detach --distribution BigInsights**
3. **rpm -e gpfs.hadoop-2-connector**
4. Remove the IBM Spectrum Scale-specific configuration from your Hadoop `core-site.xml` file. Modify the **fs.defaultFS** into an HDFS schema format by removing the following configurations:

   ```
   fs.AbstractFileSystem.gpfs.impl, fs.AbstractFileSystem.hdfs.impl, fs.gpfs.impl, fs.hdfs.impl,
   gpfs.mount.dir, gpfs.supergroup
   ```

Install and setup the HDFS transparency, see the "Upgrading HDFS transparency cluster" for more information.

Upgrading HDFS transparency cluster

This section explains how to upgrade HDFS transparency cluster.

1. Back up the configuration, in case of any failures.
2. Stop the HDFS transparency service on all nodes using the command: **/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector stop**
3. Upgrade the RPM on each node by using the command: **rpm –U gpfs.hdfs-protocol-2.7.0-<x>.x86_64.rpm**. It does not update any configuration files under the `/usr/lpp/mmfs/hadoop/etc/hadoop` folder.

   `core-site.xml`, `hdfs-site.xml`, and `slaves` files are not removed during the upgrade.
4. Start HDFS transparency service on all nodes using the command: **/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl connector start**

Limitation and difference from HDFS

This section describes the limitations and the configurations in IBM Spectrum Scale that are different from HDFS.

Property name	Value	Limitation
`dfs.permissions.enabled`	True/false	For HDFS transparency, permission check is always done.

Property name	Value	Limitation
`dfs.namenode.acls.enabled`	True/false	In native HDFS, the NameNode manages all metadata including ACL information. So, HDFS can use this to turn on or off the ACL checking. However, for IBM Spectrum Scale, HDFS transparency does not hold the metadata. When it is on, the ACL is set and stored in the Spectrum Scale file system. If the administrator turns this off later, the ACL entries set before are still stored and active in IBM Spectrum Scale.
`dfs.blocksize`	Long digital	Must be an integer multiple of the Spectrum Scale filesystem block size, **`mmlsfs -B`**. The maximum value is `1024 * file-system-data-block-size`.
`Spectrum Scale.data.dir`	String	All users of Hadoop must have full access to this directory. If this configuration is omitted, then users of Hadoop must have full access to the `Spectrum Scale.mount.dir` directory.
`dfs.namenode.fs-limits.max-xattrs-per-inode`	INT	Does not apply to the HDFS transparency.
`dfs.namenode.fs-limits.max-xattr-size`	INT	Does not apply to the HDFS transparency.

Functional limitations

1. Maximum number of extended attributes is limited by IBM Spectrum Scale. The total size of the extended attribute key and value should be less than a metadata block size in IBM Spectrum Scale.
2. Extended attributes operation on snapshots is not supported now.
3. Raw namespace is not implemented as it is not used internally.

HDFS transparency security

Configuration and binary permissions

All configuration files for HDFS transparency are located in the `/usr/lpp/mmfs/hadoop/etc/hadoop` folder after installation. Configuration files can be read and modified only by the root user.

Note: For security considerations, the root user must not grant read and write permissions to the non-root users.

The following example shows the output of the **`ls -la`** command:

```
/usr/lpp/mmfs/hadoop]# ls –la
drwx------  3 root root 4096 Nov  9 09:56 etc
```

The output of the **`ls -la`** command displays the permissions of the HDFS transparency scripts:

```
/usr/lpp/mmfs/hadoop/bin]# ls –la
-r-xrxr-x 1 root root 4484 Nov  6 10:38 gpfs

/usr/lpp/mmfs/hadoop/sbin
[root@c8f2n09 sbin]# ls –la
total 48
```

```
drwxr-xr-x  2 root root 4096 Nov 16 05:21 .
drwxr-xr-x 10 root root 4096 Nov 16 05:38 ..
-r-x------  1 root root 3310 Nov 16 05:20 deploy-gpfs.sh
-r-xr-xr-x  1 root root  697 Nov 16 05:20 gpfs-state.sh
-r-xr-xr-x  1 root root 5380 Nov 16 05:20 hadoop-daemon.sh
-r-xr-xr-x  1 root root 1360 Nov 16 05:20 hadoop-daemons.sh
-r-xr-xr-x  1 root root 4959 Nov 16 05:20 mmhadoopctl
-r-xr-xr-x  1 root root 2145 Nov 16 05:20 slaves.sh
-r-x------  1 root root 1111 Nov 16 05:20 start-gpfs.sh
-r-x------  1 root root  740 Nov 16 05:20 stop-gpfs.sh
```

The root user must keep the permissions of all the configuration files unchanged after the installation.

Note: The root user must not grant the write permission to the non-root users.

The root user must start the connector because the Java binaries check the UID of the user that starts the connector and exits when the UID does not belong to a root user. Users other than root user cannot start or stop the HDFS transparency service because the HDFS transparency binary code checks the UID of the user. If the user who starts the service is not a root user, it exits.

The non-root users can run the **mmhadoopctl connector getstate** command to view the state of the connector. The read and execute permissions of the gpfs-state.sh, hadoop-daemon.sh, hadoop-daemons.sh, and slaves.sh files can be used by the non-root users to view the state of the connector.

Note: By default, HDFS transparency installs the above scripts with the default permissions. To avoid security vulnerability, the cluster administrators must ensure that the permissions for these files are not changed.

HDFS transparency daemon UID/GID and Hadoop super groups

HDFS transparency has two types of daemons: NameNode and DataNode. Both of these daemons can only be started by the root user because certain file operations, such as setPermission and setOwner, in the Hadoop distributed file system API need root privileges.

HDFS transparency binaries exit immediately when the login credentials do not match the UID/GID of the root user. The **dfs.permissions.superusergroup** parameter in hdfs-site.xml and the **gpfs.supergroup** parameter in `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` are used by the customers to configure the Hadoop super group.

The **dfs.permissions.superusergroup** parameter can be configured as a single group and **gpfs.supergroup** can be a comma-separated group list. All users in Hadoop super groups have super privileges in the Hadoop cluster, like the super user root in Linux/Unix OS.

For non-Hadoop super users, when HDFS transparency receives RPC requests from the HDFS clients, HDFS transparency creates new threads called setfsuid/setfsgid. HDFS transparency creates these threads to replace the user ID or the group ID of the threads with the user ID or group ID of the client and handle the requests. This can restrict the privileges of a common user in the Hadoop cluster.

For Hadoop super users, all operations are performed under the security context of the root user. The user must configure Hadoop super groups carefully.

The simple security mode

When Kerberos is not enabled, Hadoop runs under the simple security mode. In this mode, RPCs are not encrypted and authenticated, and all users can submit maps and reduce jobs to the Hadoop cluster. A Hadoop cluster running in the simple security mode is vulnerable to attack via the network from outside the clusters and from users logged on to the nodes in the cluster.

Note: You must enable Kerberos. For more information about Kerberos, see "The Kerberos mode." The data transfers and RPCs from the clients to the NameNode and DataNode are not encrypted, and therefore, vulnerable to attack through the network.

In the hadoop cluster, the user ID must be created on all the nodes including the nodes submitting jobs and the nodes running NameNode and DataNode. If a user submits the jobs, but the user ID is not created in DataNodeX, Map and Reduce jobs on DataNodeX cannot access the data because Hadoop creates the files with the user ID and group ID of the user. Hadoop itself does not manage user IDs and group IDs. Hadoop only transfers the user ID and group ID from the job submitter and stores them as the owner of the job output file in the file system. For a user to read or write a file, permissions need to be set using the traditional Linux/Unix permission control. The authentication of a user is done using Linux authentication. If a user has successfully logged on to the system, the user has passed the OS authentication,

The **fs.permissions.umask-mode** parameter in hdfs-site.xml can be configured as the umask used while creating files and directories. For more information about this configuration, see Hadoop website.

For more information about security configurations, see HDFS Permission Guide and HDFS Permissions and Security Guide.

ACL:

The **dfs.namenode.acls.enabled** property in hdfs-site.xml can be used to enable support for ACLs by HDFS transparency.

Note: Hadoop only supports POSIX ACL. If the applications set NFS ACL for certain files through the POSIX interface, jobs fail while handling the ACL of those files and java exceptions are reported in the GPFS HDFS transparency logs.

Namenode block access token:

In the previous releases of Hadoop, DataNode did not enforce access control on the data blocks. If an unauthorized client provided the block ID, the client could read a data block. Also, unauthorized users were able to write data blocks to DataNodes.

In Hadoop Release 0.2x and later, for HDFS transparency, when clients request to access files, the file permissions are checked. Only if the client has the required permissions, NameNode returns a token in the HMAC-SHA1 format to the client. The client sends the token back to DataNode when it requests data access. DataNode checks this token and grants or refuses access to the block.

To enable the NameNode block access token, configure the following settings in the hdfs-site.xml file:

```
dfs.block.access.token.enable=yes
dfs.block.access.key.update.interval=600 (by default, minutes)
dfs.block.access.token.lifetime=600 (by default, minutes)
```

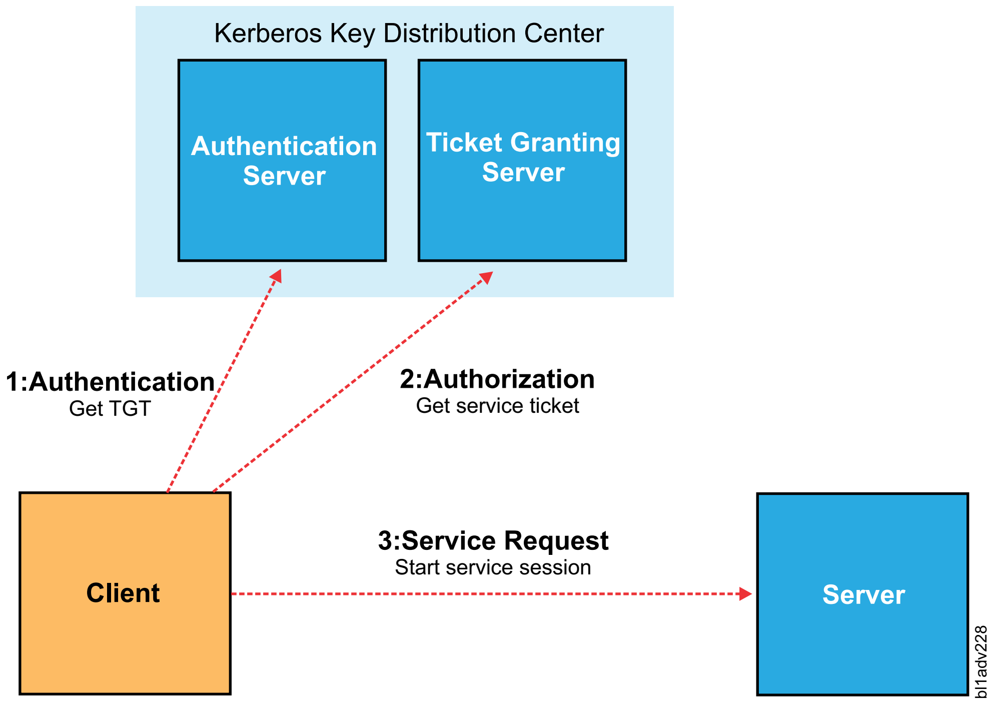
Note: By default, this feature is enabled in the IBM BigInsight IOP distribution. However, this feature cannot prevent the attacker from connecting to NameNode if Kerberos is not enabled.

The Kerberos mode

User authentication and authorization is weak in the simple mode. The data transfers and RPCs from the clients to the NameNode and DataNode are not encrypted. The Kerberos mode introduced in the Hadoop ecosystem provides a secure Hadoop environment.

The Kerberos service comprises of a client-server architecture that provides secure transactions over networks. The service offers strong user authentication, as well as integrity and privacy. The authentication verifies the identities of the sender and the receiver in a network transaction. The service also checks for data integrity and encrypts the data during transmission.

Using the Kerberos service, you can log on to other machines, execute commands, exchange data, and transfer files securely. Additionally, Kerberos provides authorization services that allow administrators to restrict access to services and machines.



So, in the Kerberos mode, only authorized users can access services, thereby preventing an unauthorized access to services. The Kerberos mode also encrypts the data during transmission to avoid data exposure.

To enable Kerberos, configure the core-site.xml as follows:

```
hadoop.security.authorization=true
hadoop.security.authentication=Kerberos (the default is "simple")
```

SASL/GSS API and RPC:

This topic describes the server-side authentication, client-side authentication, and Hadoop client, Hadoop server, and RPC

Server-side authentication

Hadoop services, such as NameNode and DataNode, in HDFS transparency must authenticate to Kerberos KDC. During the startup, the service will log in KDC by using the service principal and the keytab configured in the core-site.xml file.

Note: All keytab files used by Hadoop services are stored in a local file system of the node running the services. Different Hadoop distro might take different locations for this. For IBM BigInsights IOP, the distro will take `/etc/security/keytabs/` on the nodes that are running the services. Different keytab files are owned by different users and are readable only by the owner of the file. Hadoop cluster administrator must be careful and must not expose the read and write permission of these files to other users.

After the service authentication check passes, the service finishes the start-up procedure and is ready to handle the client requests.

Client-side authentication

A Hadoop user must be authenticated by the Kerberos KDC before accessing the Hadoop services through the client tool by using their own user principal.

The steps for Hadoop client to submit jobs:

1. Log on to a client machine that is connected to the Hadoop cluster, and then execute the **`kinit`** command with the principal and the password.
2. The **`kinit`** command authenticates the user with the KDC, gets the Kerberos TGT ticket, and puts the ticket into the ticket cache in the local file system.
3. Run the client tools. For example, submit a MapReduce job through the JobClient.

The client bootstraps and issues connection requests to the server side.

Hadoop client, Hadoop server, and RPC

After the server and client sides authenticate with Kerberos successfully, the server waits for the client requests. After the client issues a request, both server and client come down to the SASL/GSSAPI stack:

1. The client stack picks up the client TGT ticket in the current access control context.
2. Using the TGT, the client requests a service ticket from the KDC targeting the right service or server that the user or the client software is accessing.
3. The client sends out the service ticket first as part of the connector with the service. The server or service decrypts the service ticket with the service key. The service provides the service key when it authenticates with KDC. If the server can decrypt the service ticket successfully, it means that the client has passed the authentication.

The workflow in the SASL/GSSAPI stack regarding SASL and GSSAPI specifications involving Kerberos is complex, but it is not just for authentication, as it also builds a secure context and channel for both the server and client sides.

Three levels of protection are provided by this stack: auth (authentication), int (integrity), and privacy (encryption). These options are exposed and can be configured in the latest version of Hadoop making the encryption of the RPC channel easy.

This feature is controlled by **`hadoop.rpc.protection`** in core-site.xml. The authentication value is for enabling SASL connections for authentication, the integrity value is for enabling SASL connections for authentication and data integrity, and the privacy value is for enabling SASL connections for authentication, data integrity, and privacy (encrypting the data).

Delegated NameNode token:

All operations from client nodes must be connected with the KDC to be authenticated when Kerberos is enabled. This process can impact the performance of the Hadoop jobs. Therefore, the NameNode Token delegation was introduced to reduce the performance impact from Kerberos and the load on the KDC server.

Authenticating clients through delegated NameNode tokens is a two-way authentication protocol that is based on Java SASL Digest-MD5. The token is obtained during job submissions, and then submitted to JobTracker. The steps are as follows:

1. The user authenticates the JobTracker by using Kerberos.

2. By using Kerberos, the user authenticates the NameNode(s) that the tasks will interact with at runtime. The user gets a delegation token from each of the NameNodes.
3. The user passes the tokens to the JobTracker as part of the job submission.

All TaskTrackers running the job tasks get a copy of the tokens through an HDFS location that is private to the user that the MapReduce daemons run. The tokens are written to a file in a private area that is visible to the job-owner user on the TaskTracker machine.

While launching the task, the TaskTracker exports the location of the token file as an environment variable. The task process loads the tokens into the memory. The file is read as part of the static initialization of the `UserGroupInformation` class used in the Hadoop services. This information is useful for the RPC client.

In the Kerberos mode, the Apache Hadoop RPC client can communicate securely with a server by using either tokens or Kerberos. The RPC client is programmed in a way that when a token exists for a service, it will be used for secure communication. If no token is available, Kerberos is used.

When Kerberos is enabled, Delegated NameNode token takes effect automatically. The configurations settings related to this feature are in the hdfs-site.xml file:

```
dfs.namenode.delegation.key.update-interval(milliseconds)
dfs.namenode.delegation.token.max-lifetime(milliseconds)
dfs.namenode.delegation.token.renew-interval(milliseconds)
```

HTTP SPNEGO authentication:

By default, Hadoop web applications such as ResourceManager, NodeNodeManager, JobTracker, NameNode, TaskTrackers, and DataNodes can be accessed without authentication. If Kerberos is enabled, all web applications can be configured to authenticate through Kerberos HTTP SPNEGO.

The configurations settings for this feature can be viewed in core-site.xml and the following properties values must be changed:

```
<!--  HTTP web-consoles Authentication -->
  <property>
<name>hadoop.http.filter.initializers</name>

<value>org.apache.hadoop.security.AuthenticationFilterInitializer</value>
  </property>

  <property>
  <name>hadoop.http.authentication.type</name>
  <value>kerberos</value>
  </property>

  <property>
  <name>hadoop.http.authentication.token.validity</name>
  <value>36000</value>
  </property>

  <property>
  <name>hadoop.http.authentication.signature.secret.file</name>
  <value>/hadoop/hadoop/conf/http-secret-file</value>
  </property>

  <property>
  <name>hadoop.http.authentication.cookie.domain</name>
  <value></value>
  </property>

  <property>
  <name>hadoop.http.authentication.simple.anonymous.allowed</name>
  <value>false</value>
```

```
</property>

<property>
<name>hadoop.http.authentication.kerberos.principal</name>
<value>HTTP/hz169-91.i.site.com@I.NETEASE.COM</value>
</property>

<property>
<name>hadoop.http.authentication.kerberos.keytab</name>
<value>/hadoop/hadoop/conf/http.keytab</value>
</property>
```

RPC and data encryption:

To encrypt data that is transferred between Hadoop services and clients, set **hadoop.rpc.protection** to privacy in core-site.xml.

To activate data encryption for the data transfer protocol of DataNode, set **dfs.encrypt.data.transfer** to true in hdfs-site.xml. Optionally, set **dfs.encrypt.data.transfer.algorithm** to either 3DES or RC4 to choose the specific encryption algorithm. If the encryption algorithm is not specified then the default value configured for JCE, which is usually 3DES, is used for the system. Setting **dfs.encrypt.data.transfer.cipher.suites** to AES/CTR/NoPadding activates AES encryption. By default, this is not specified. Therefore, AES is not used. When AES is specified, the algorithm specified in **dfs.encrypt.data.transfer.algorithm** is still used during the initial key exchange. The AES key bit length can be configured by setting **dfs.encrypt.data.transfer.cipher.key.bitlength** to 128, 192, or 256. The default value is 128.

AES offers the greatest cryptographic strength and the best performance. At this time, 3DES and RC4 are most commonly used in Hadoop clusters.

Data transfers between Web console and clients are protected using SSL (HTTPS), such as httpfs and webHDFS.

If your Hadoop cluster has to be NIST-compliant, you must select NIST-compliant encryption algorithm and key length. For NIST-compliant algorithm and key length, see Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths. Also, you must configure the IBM Spectrum Scale cluster as NIST-compliant by running the **mmchconfig nistCompliance=SP800-131A** command in the IBM Spectrum Scale cluster.

Note: 3DES and AES are NIST-compliant whereas RC4 is not NIST-compliant.

Shortcircuit and security

In HDFS, shortcircuit can be enabled when a client and DataNode are on the same node. By enabling shortcircuit, an application that needs to read a file can obtain the file descriptor from the DataNode and read the data block directly. Shortcircuit reads provide a significant boost in the read I/O performance. The Hadoop client can only read data from the file descriptor because the DataNode opens the file in the read-only mode.

For HDFS transparency with FPO configuration, this feature can be enabled for enhanced performance. In the shortcircuit mode, the DataNode and the client communicate through a Unix domain socket that is configured through `dfs.domain.socket.path` in hdfs-site.xml, configured as `/var/lib/hadoop-hdfs/dn_socket`.

```
/var/lib/hadoop-hdfs]# ls -l
drwxr-xr-x  2 root root  4096 Nov  5 01:04 hadoop-hdfs
srw-rw-rw-  1 root root     0 Nov  5 01:04 dn_socket
```

The permission for the socket file must be 666 so that all common users can read the socket and receive messages from the DataNode. The x permission bit does not matter here

When Kerberos is enabled, the Kerberos server authenticates the Hadoop client and the DataNode checks the authorization of the Hadoop client by checking the service ticket. Whether Kerberos is enabled or not, the DataNode checks the Block Access Token from the Hadoop client and ensures that the Hadoop client can access the target file before the file descriptor is sent to the Hadoop client. These checks ensure that the file descriptor is not sent to invalid users.

Note: The **`dfs.block.access.token.enable`** parameter must be configured as true when shortcircuit is enabled.

Also, the message transfer over the Unix domain socket in shortcircuit is not encrypted. For more information about security considerations in shortcircuit, see HDFS 5353. However, as the data is transferred over the same machine and not over the TCP/IP network, the message transfer is considered safe. To avoid security vulnerabilities from shortcircuit, disable the feature.

Hadoop data isolation

This topic describes Hadoop data isolation.

Hadoop super users can control the data in the file system. If you do not want the Hadoop super users to access the data in the POSIX applications, configure `gpfs.data.dir` in `/usr/lpp/mmfs/hadoop/etc/hadoop/gpf-site.xml` to isolate the Hadoop data under `/<gpfs.mnt.dir>/<gpfs.data.dir>/`. This configuration setting ensures that the Hadoop super users can only access the data in the `/<gpfs.mnt.dir>/<gpfs.data.dir>/` folder. Data outside the `/<gpfs.mnt.dir>/<gpfs.data.dir>/` folder cannot be accessed by the Hadoop super users.

Hadoop data access audit

This section describes the Hadoop data access audit.

HDFS Transparency is certified with IBM Security Guardium DAM (Database Activity Monitoring) to monitor the Hadoop data access over IBM Spectrum Scale.

For more information about Hadoop data access audit, see Big data security and auditing with IBM InfoSphere Guardium.

Guide for security setup

Enable Kerberos for IBM BigInsights IOP

For manual HDFS replacement mode:

This topic lists the steps to enable Kerberos for IBM BigInsights IOP for manual HDFS replacement mode.

In this mode, users must install IOP over HDFS, and then replace HDFS with IBM Spectrum Scale.

If you use this mode to deploy IOP and IBM Spectrum Scale, perform the following steps to enable Kerberos:

1. To set up the KDC server, go to Setting up a KDC manually.
2. Shut down the GPFS service and start the HDFS service.

   Note: For the FPO model and the shared storage model, HDFS transparency nodes must be part of the IOP cluster. IOP services do not start when the NameNode service is running over the node outside the IOP cluster.
3. In the Ambari GUI, click **Admin** > **Kerberos**, and follow the guide to enable the Kerberos service.

   Note: In the GUI wizard, select existing MIT KDC and type the required input according to the configuration.

4. Run a service check for all the services. For a user who has not authenticated with the KDC, the system reports a failure:

```
[fvtest@c8f2n06 ~]$ yarn org.apache.hadoop.yarn.applications.distributedshell.Client
-shell_command ls -num_containers 3 -jar
/usr/iop/current/hadoop-yarn-client/hadoop-yarn-applications-distributedshell.jar
15/11/02 02:51:46 INFO distributedshell.Client: Initializing Client
15/11/02 02:51:46 INFO distributedshell.Client: Running Client
15/11/02 02:51:46 INFO client.RMProxy: Connecting to ResourceManager at
c8f2n07.gpfs.net/192.168.105.163:8050
15/11/02 02:51:47 WARN ipc.Client: Exception encountered while connecting to the
server : javax.security.sasl.SaslException: GSS initiate failed [Caused by GSSException:
No valid credentials provided (Mechanism level: Failed to find any Kerberos tgt)]
15/11/02 02:51:47 FATAL distributedshell.Client: Error running Client
java.io.IOException: Failed on local exception: java.io.IOException:
javax.security.sasl.SaslException: GSS initiate failed
[Caused by GSSException: No valid credentials provided (Mechanism level:
Failed to find any Kerberos tgt)]; Host Details : local host is:
"c8f2n06/192.168.105.162"; destination host is: "c8f2n07.gpfs.net":8050;
        at org.apache.hadoop.net.NetUtils.wrapException(NetUtils.java:773)
        at org.apache.hadoop.ipc.Client.call(Client.java:1480)
        at org.apache.hadoop.ipc.Client.call(Client.java:1407)
        at org.apache.hadoop.ipc.ProtobufRpcEngine$Invoker.invoke(ProtobufRpcEngine.java:229)
        at com.sun.proxy.$Proxy7.getClusterMetrics(Unknown Source)
        at org.apache.hadoop.yarn.api.impl.pb.client.ApplicationClientProtocolPBClientImpl.getClusterMetrics
        (ApplicationClientProtocolPBClientImpl.java:206)
```

5. Shut down the HDFS service.
6. In Ambari GUI, click **HDFS** > **configs** in the Advanced tag page, and add the following to Custom core-site:

```
hadoop.proxyuser.yarn.groups=*
hadoop.proxyuser.yarn.hosts=*
```

   If HTTPS is not configured, the user must ensure that `dfs.http.policy` is HTTP_ONLY and `dfs.https.enable` is false. Otherwise, the IOP services will not start.
7. Install the gpfs.hdfs-protocol rpm.
8. On any one IOP node, run the **`mmhadoopctl connector syncconf /etc/hadoop/conf`** command. This synchronizes all the Hadoop configurations into the HDFS transparency configuration directory.

   Note: The **`/etc/hadoop/conf`** parameter is the Hadoop configuration directory for IOP.
9. Modify `/usr/lpp/mmfs/hadoop/etc/hadoop/gpfs-site.xml` according to your cluster.
10. On the same node, run:

```
/usr/lpp/mmfs/hadoop/sbin/deploy-gpfs.sh –nocheck /usr/lpp/mmfs/hadoop/etc/hadoop/
/usr/lpp/mmfs/hadoop/etc/hadoop/
```

11. For an FPO model, run the following commands on any one node:

```
mmdsh –N all "chown root:root /etc/security/keytabs/dn.service.keytab"
mmdsh -N all "chown root:root /var/lib/hadoop-hdfs"
```

   For the shared storage model, run the following commands on any one node that is running the HDFS transparency service:

```
mmdsh –N all "chown root:root /etc/security/keytabs/dn.service.keytab"
mmdsh -N all "chown root:root /var/lib/hadoop-hdfs"
```

   The `/usr/lpp/mmfs/hadoop/sbin/mmhadoopctl` connector starts.
12. Go to the IOP Ambari GUI to start the other services.

   Note: If shortcircuit is enabled, `dfs.domain.socket.path=/var/lib/hadoop-hdfs/dn_socket` must be owned by `root:root`. If it is not, the DataNode service will not start.

Automatic GPFS deployment with IOP:

In Release gpfs.hdfs-protocol.2.7.0-1, HDFS transparency is not integrated with BigInsights IOP.

Enable the HTTPS service of NameNode:

This topic lists the steps to enable the HTTPS service.

By default, the HTTPS service is not enabled. If you want to enable the HTTPS service, perform the following steps:

1. Generate the key and the certificate. To deploy HTTPS, a key and certificate must be generated for each machine in the cluster. You can use Java's keytool utility to accomplish this task:

   ```
   $ keytool -keystore {keystore} -alias localhost -validity {validity} –genkey
   ```

   The parameter definitions are as following:

   - keystore: The keystore file that stores the certificate. The keystore file contains the private key of the certificate, and must be kept safely.
   - validity: The validity of the certificate in days.

   The keytool needs more details of the certificate, such as the hostname and the organization name.

   Note: The hostname (CN) is the hostname of the HDFS Transparency NameNode.
2. Create your own CA. Each machine in the cluster has a public-private key pair, and a certificate to identify the machine. The certificate, however, is unsigned, which means that an attacker can create such a certificate and become an authorized user. Use `openssl` to generate a new CA certificate:

   ```
   openssl req -new -x509 -keyout <ca-key> -out <ca-cert> -days <validity>
   ```

   The generated CA is simply a public-private key pair and certificate and is intended to sign other certificates.
3. Add the generated CA to the client truststore.

   ```
   $ keytool -keystore {truststore} -alias CARoot -import -file {ca-cert}
   ```

   In contrast to the keystore that stores the machine identity, the truststore of the client stores all the certificates that the client must trust. Importing a certificate into a truststore means that the client trusts all the certificates that are signed by that certificate. This attribute is called the chain of trust, and it is particularly useful while deploying HTTPS on a large Hadoop cluster. You can sign all the certificates in the cluster with a single CA, and have all machines share the same truststore that trusts the CA. The machines can then authenticate all other machines.
4. Sign all the generated certificates and the CA. Perform the following steps:
   a. Export the certificate from the keystore.

      ```
      $ keytool -keystore -alias localhost -certreq -file {cert-file}
      ```

   b. Sign the certificate with the CA.

      ```
      $ openssl x509 -req -CA {ca-cert} -CAkey {ca-key} -in {cert-file}
      -out {cert-signed} -days {validity} -CAcreateserial -passin pass:{ca-password}
      ```

   c. Import the CA certificate and the signed certificate into the keystore.

      ```
      $ keytool -keystore -alias CARoot -import -file {ca-cert}
      $ keytool -keystore -alias localhost -import -file {cert-signed}
      ```

   The parameter definitions are as follows:

   - keystore: the location of the keystore
   - ca-cert: the certificate of the CA
   - ca-key: the private key of the CA
   - ca-password: the passphrase of the CA
   - cert-file: the exported, unsigned certificate of the server
   - cert-signed: the signed certificate of the server
5. Configure the HDFS transparency NameNode in the hdfs-site.xml file:

   ```
   <property>
   <name>dfs.http.policy</name>
   <value>HTTP_AND_HTTPS</value>
   ```

```
</property>

<property>
<name>dfs.https.enable</name>
<value>true</value>
</property>
```

The **dfs.http.policy** parameter can be one of the following:

- HTTP_ONLY: Only the HTTP server has started.
- HTTPS_ONLY: Only the HTTPS server has started.
- HTTP_AND_HTTPS: The HTTP and HTTPS servers have started.

Note: If you configure the **dfs.http.policy** parameter as HTTPS_ONLY or HTTP_AND_HTTPS, **webhdfs** of HDFS transparency NameNode becomes unavailable. For more information about applications requiring **swebhdfs**, see HDFS-3987.

Note: The **swebhdfs** parameter is not available for Hadoop Release 2.3.0 and earlier. You must consider upgrading your Hadoop release version for enhanced security.

6. Configure ssl-server.xml as:

```
<property>
<name>ssl.server.keystore.type</name>
<value>jks</value>
</property>
<property>
<name>ssl.server.keystore.keypassword</name>
<value><password of keystore></value>
</property>
<property>
<name>ssl.server.keystore.location</name>
<value><location of keystore.jks></value>
</property>
<property>
<name>ssl.server.truststore.type</name>
<value>jks</value>
</property>
<property>
<name>ssl.server.truststore.location</name>
<value><location of truststore.jks></value>
</property>
<property>
<name>ssl.server.truststore.password</name>
<value><password of truststore></value>
</property>
```

Also, configure the ssl-client.xml as:

```
<property>
<name>ssl.client.truststore.password</name>
<value><password of truststore></value>
</property>

<property>
<name>ssl.client.truststore.type</name>
<value>jks</value>
</property>
<property>
<name>ssl.client.truststore.location</name>
<value><location of truststore.jks></value>
</property>
```

7. To restart the HDFS transparency services, run the **mmhadoopctl** command.

Note: Remember to sync hdfs-site.xml, ssl-server.xml, and ssl-client.xml from BigInsights IOP, `/etc/hadoop/conf`, with the HDFS transparency configuration directory, `/usr/lpp/mmfs/hadoop/etc/hadoop`, for all nodes that are running HDFS transparency.

Security configuration in Hadoop

This topic describes the security configuration in Hadoop.

The Kerberos-related configuration changes in the hdfs-site.xml file are restricted to HDFS transparency. However, enabling Kerberos impacts the other Hadoop components as well. Therefore, other components must also be configured for Kerberos.

If changes are made only to the hdfs-site.xml file, which is the configuration file used by HDFS transparency, the other Hadoop services fail.

Add the following configuration settings to the core-site.xml file:

```
<property>
 <name>hadoop.http.authentication.cookie.domain</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.cookie.path</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.kerberos.name.rules</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.signature.secret</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.signature.secret.file</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.signer.secret.provider</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.signer.secret.provider.object</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.token.validity</name>
 <value></value>
</property>

<property>
 <name>hadoop.http.authentication.type</name>
 <value>simple</value>
</property>

<property>
 <name>hadoop.http.filter.initializers</name>
 <value></value>
</property>

<property>
 <name>hadoop.proxyuser.HTTP.groups</name>
 <value>users</value>
```

```
  </property>

  <property>
   <name>hadoop.proxyuser.HTTP.hosts</name>
   <value>c8f2n07.gpfs.net</value>
  </property>

  <property>
   <name>hadoop.proxyuser.knox.groups</name>
   <value>users</value>
  </property>

  <property>
   <name>hadoop.proxyuser.knox.hosts</name>
   <value>c8f2n06.gpfs.net</value>
  </property>

  <property>
   <name>hadoop.rpc.protection</name>
   <value>authentication</value>
  </property>

  <property>
   <name>hadoop.security.auth_to_local</name>
   <value>RULE:[1:$1@$0](ambari-qa@gpfs.net)s/.*/ambari-qa/
RULE:[1:$1@$0](hbase@gpfs.net)s/.*/hbase/
RULE:[1:$1@$0](hdfs@gpfs.net)s/.*/hdfs/
RULE:[1:$1@$0](spark-08212047@gpfs.net)s/.*/spark/
RULE:[1:$1@$0](.*@gpfs.net)s/@.*//
RULE:[2:$1@$0](HTTP@gpfs.net)s/.*/hbase/
RULE:[2:$1@$0](amshbase@gpfs.net)s/.*/ams/
RULE:[2:$1@$0](dn@gpfs.net)s/.*/hdfs/
RULE:[2:$1@$0](hbase@gpfs.net)s/.*/hbase/
RULE:[2:$1@$0](hive@gpfs.net)s/.*/hive/
RULE:[2:$1@$0](jhs@gpfs.net)s/.*/mapred/
RULE:[2:$1@$0](jn@gpfs.net)s/.*/hdfs/
RULE:[2:$1@$0](knox@gpfs.net)s/.*/knox/
RULE:[2:$1@$0](nfs@gpfs.net)s/.*/hdfs/
RULE:[2:$1@$0](nm@gpfs.net)s/.*/yarn/
RULE:[2:$1@$0](nn@gpfs.net)s/.*/hdfs/
RULE:[2:$1@$0](oozie@gpfs.net)s/.*/oozie/
RULE:[2:$1@$0](rm@gpfs.net)s/.*/yarn/
RULE:[2:$1@$0](solr@gpfs.net)s/.*/solr/
RULE:[2:$1@$0](yarn@gpfs.net)s/.*/yarn/
RULE:[2:$1@$0](zookeeper@gpfs.net)s/.*/ams/
RULE:[2:$1@$0]([nd]n@.*)s/.*/hdfs/
RULE:[2:$1@$0]([rn]m@.*)s/.*/yarn/
RULE:[2:$1@$0](hm@.*)s/.*/hbase/
RULE:[2:$1@$0](jhs@.*)s/.*/mapred/
RULE:[2:$1@$0](rs@.*)s/.*/hbase/
DEFAULT</value>
  </property>

  <property>
   <name>hadoop.security.authentication</name>
   <value>kerberos</value>
  </property>

  <property>
   <name>hadoop.security.authorization</name>
   <value>true</value>
  </property>
```

Add the following configuration settings in the hdfs-site.xml file. Even if there is a single property, modify the property:

```
<property>
 <name>dfs.datanode.address</name>
 <value>0.0.0.0:1019</value>
</property>

<property>
 <name>dfs.datanode.http.address</name>
 <value>0.0.0.0:1022</value>
</property>

<property>
 <name>dfs.datanode.kerberos.principal</name>
 <value>dn/_HOST@gpfs.net</value>
</property>

<property>
 <name>dfs.namenode.kerberos.internal.spnego.principal</name>
 <value>HTTP/_HOST@gpfs.net</value>
</property>

<property>
 <name>dfs.namenode.kerberos.principal</name>
 <value>nn/_HOST@gpfs.net</value>
</property>

<property>
 <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
 <value>HTTP/_HOST@gpfs.net</value>
</property>

<property>
 <name>dfs.secondary.namenode.kerberos.principal</name>
 <value>nn/_HOST@gpfs.net</value>
</property>

<property>
 <name>dfs.web.authentication.kerberos.principal</name>
 <value>HTTP/_HOST@gpfs.net</value>
</property>

<property>
 <name>nfs.kerberos.principal</name>
 <value>nfs/_HOST@gpfs.net</value>
</property>

<property>
 <name>nfs.keytab.file</name>
 <value>/etc/security/keytabs/nfs.service.keytab</value>
</property>

<property>
 <name>dfs.http.policy</name>
 <value>HTTP_AND_HTTPS</value>
</property>

<property>
 <name>dfs.https.enable</name>
 <value>true</value>
</property>
```

The following is the modification applied to the mapred-site.xml file:

Note: If the property exists, modify the property.

```
<property>
 <name>mapreduce.jobhistory.keytab</name>
 <value>/etc/security/keytabs/jhs.service.keytab</value>
```

```
</property>

<property>
 <name>mapreduce.jobhistory.principal</name>
 <value>jhs/_HOST@gpfs.net</value>
</property>

<property>
 <name>mapreduce.jobhistory.webapp.spnego-keytab-file</name>
 <value>/etc/security/keytabs/spnego.service.keytab</value>
</property>

<property>
 <name>mapreduce.jobhistory.webapp.spnego-principal</name>
 <value>HTTP/_HOST@gpfs.net</value>
</property>
```

The following is the modification applied to the yarn-site.xml file:

Note: If the property exists, modify the property.

```
<property>
 <name>yarn.acl.enable</name>
 <value>true</value>
</property>

<property>
 <name>yarn.nodemanager.container-executor.class</name>
 <value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</value>
</property>

<property>
 <name>yarn.nodemanager.keytab</name>
 <value>/etc/security/keytabs/nm.service.keytab</value>
</property>

<property>
 <name>yarn.nodemanager.linux-container-executor.cgroups.mount-path</name>
 <value></value>
</property>

<property>
 <name>yarn.nodemanager.principal</name>
 <value>nm/_HOST@gpfs.net</value>
</property>

<property>
 <name>yarn.nodemanager.webapp.spnego-keytab-file</name>
 <value>/etc/security/keytabs/spnego.service.keytab</value>
</property>

<property>
 <name>yarn.nodemanager.webapp.spnego-principal</name>
 <value>HTTP/_HOST@gpfs.net</value>
</property>

<property>
 <name>yarn.resourcemanager.keytab</name>
 <value>/etc/security/keytabs/rm.service.keytab</value>
</property>

<property>
 <name>yarn.resourcemanager.principal</name>
 <value>rm/_HOST@gpfs.net</value>
</property>

<property>
```

```
 <name>yarn.resourcemanager.proxy-user-privileges.enabled</name>
 <value>true</value>
</property>

<property>
 <name>yarn.resourcemanager.proxyusers.*.groups</name>
 <value></value>
</property>

<property>
 <name>yarn.resourcemanager.proxyusers.*.hosts</name>
 <value></value>
</property>

<property>
 <name>yarn.resourcemanager.proxyusers.*.users</name>
 <value></value>
</property>

<property>
 <name>yarn.resourcemanager.webapp.spnego-keytab-file</name>
 <value>/etc/security/keytabs/spnego.service.keytab</value>
</property>

<property>
 <name>yarn.resourcemanager.webapp.spnego-principal</name>
 <value>HTTP/_HOST@gpfs.net</value>
</property>

<property>
 <name>yarn.timeline-service.enabled</name>
 <value>false</value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.cookie.domain</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.cookie.path</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.kerberos.keytab</name>
 <value>/etc/security/keytabs/spnego.service.keytab</value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.kerberos.name.rules</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.kerberos.principal</name>
 <value>HTTP/_HOST@gpfs.net</value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.proxyusers.*.groups</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.proxyusers.*.hosts</name>
 <value></value>
```

```
</property>

<property>
 <name>yarn.timeline-service.http-authentication.proxyusers.*.users</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.signature.secret</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.signature.secret.file</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.signer.secret.provider</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.signer.secret.provider.object</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.token.validity</name>
 <value></value>
</property>

<property>
 <name>yarn.timeline-service.http-authentication.type</name>
 <value>kerberos</value>
</property>

<property>
 <name>yarn.timeline-service.keytab</name>
 <value>/etc/security/keytabs/yarn.service.keytab</value>
</property>

<property>
 <name>yarn.timeline-service.principal</name>
 <value>yarn/_HOST@gpfs.net</value>
</property>
```

Chapter 19. Encryption

GPFS provides support for file encryption that ensures both secure storage and secure deletion of data. GPFS manages encryption through the use of encryption keys and encryption policies.

Note: GPFS encryption is only available with IBM Spectrum Scale Advanced Edition. The file system must be at the latest version for GPFS 4.1. Encryption is supported in multicluster environments (provided that the remote nodes have their own `/var/mmfs/etc/RKM.conf` files and access to the remote key managers; see "Encryption keys") and FPO environments.

Secure storage uses encryption to make data unreadable to anyone who does not possess the necessary encryption keys. The data is encrypted while "at rest" (on disk) and is decrypted on the way to the reader. Only data, not metadata, is encrypted.

GPFS encryption can protect against attacks targeting the disks (for example, theft or acquisition of improperly discarded disks) as well as attacks performed by unprivileged users of a GPFS node in a multi-tenant cluster (that is, a cluster that stores data belonging to multiple administrative entities called tenants). However, it cannot protect against deliberate malicious acts by a cluster administrator.

Secure data deletion leverages encryption and key management to guarantee erasure of files beyond the physical and logical limitations of normal deletion operations. If data is encrypted, and the master key (or keys) required to decrypt it have been deleted from the key server, that data is effectively no longer retrievable. See "Encryption keys."

Important: Encryption should not be viewed as a substitute for using file permissions to control user access.

For more information on configuring encryption, see the Encryption chapter in the IBM Spectrum Scale Redbook.

Encryption keys

GPFS uses the following types of encryption keys:

master encryption key (MEK)
An MEK is used to encrypt file encryption keys.

MEKs are stored in remote key management (RKM) servers and are cached by GPFS components. GPFS receives information about the RKM servers in a separate `/var/mmfs/etc/RKM.conf` configuration file. Encryption rules present in the encryption policy define which MEKs should be used, and the `/var/mmfs/etc/RKM.conf` file provides a means of accessing those keys. The `/var/mmfs/etc/RKM.conf` also specifies how to access RKMs containing MEKs used to encrypt files created under previous encryption policies.

An MEK is identified with a unique *Keyname* that combines the name of the key and the RKM server on which it resides. See "Encryption policy rules" on page 368 for *Keyname* format.

file encryption key (FEK)
An FEK is used to encrypt sectors of an individual file. It is a unique key that is randomly generated when the file is created. For protection, it is encrypted (or "wrapped") with one or more MEKs and stored in the `gpfs.Encryption` extended attribute of the file.

A wrapped FEK cannot be decoded without access to the MEK (or MEKs) used to wrap it. Therefore, a wrapped FEK is useless to an attacker and does not require any special handling at object deletion time. If necessary, an FEK can be rewrapped using a new set of MEKs to allow for operations like MEK expiration and rotation, compromised key removal, and data expiration.

Note: If an encryption policy specifies that an FEK be wrapped multiple times, only one of the wrapped-FEK instances needs to be unwrapped for the file to be accessible.

Encryption policies

GPFS uses encryption policies to manage aspects of how file encryption is to be implemented, including the following:

- which files are to be encrypted
- which algorithm is to be used for the encryption
- which MEK (or MEKs) are to be used to wrap the FEK of a file

Encryption policies are configured using the **mmchpolicy** command and are applied at file creation time. When a file is created, encryption rules are traversed in order until one of the following occurs:

- The last rule is reached.
- The maximum number of **SET ENCRYPTION** rules that can be matched (eight) is reached.
- An **ENCRYPTION EXCLUDE** rule is matched.

If the file matches at least one **SET ENCRYPTION** rule, an FEK is generated and used to encrypt its contents. The FEK is wrapped once for each policy it matches, resulting in one or more versions of the encrypted FEK being stored in the `gpfs.Encryption` extended attribute of the file.

Notes:

1. When an encryption policy is changed, the changes apply only to the encryption of subsequently created files.
2. Encryption policies are defined on a per–file system basis by a system administrator. Once the encryption policies are put in place, they may result in files in different filesets or with different names being encrypted differently.

Encryption policy rules

GPFS provides the following rules with which you can specify encryption policies:

ENCRYPTION IS

This rule is used to specify how a file is to be encrypted and how the FEK is to be wrapped.

The syntax of the **ENCRYPTION IS** rule is:

```
RULE 'RuleName' ENCRYPTION 'EncryptionSpecificationName' IS
    ALGO 'EncParamString'
    COMBINE 'CombineParamString'
    WRAP 'WrapParamString'
    KEYS('Keyname'[, 'Keyname', ... ])
```

where:

`ALGO` *EncParamString*

specifies the encryption parameter string, which defines the following:

- encryption algorithm
- key length
- mode of operation
- key derivation function

The following encryption parameter strings are valid:

Table 55. Valid EncParamString values

Value	Description
`AES:128:XTS:FEK:HMACSHA512`	Encrypt the file with AES in XTS mode. The FEK is 128 bits long and is preprocessed using HMAC with SHA-512.
`AES:256:XTS:FEK:HMACSHA512`	Encrypt the file with AES in XTS mode. The FEK is 256 bits long and is preprocessed using HMAC with SHA-512.
`AES:128:CBC:FEK:HMACSHA512`	Encrypt the file with AES in CBC mode. The FEK is 128 bits long and is preprocessed using HMAC with SHA-512.
`AES:192:CBC:FEK:HMACSHA512`	Encrypt the file with AES in CBC mode. The FEK is 192 bits long and is preprocessed using HMAC with SHA-512.
`AES:256:CBC:FEK:HMACSHA512`	Encrypt the file with AES in CBC mode. The FEK is 256 bits long and is preprocessed using HMAC with SHA-512.

COMBINE *CombineParamString*
specifies a string that defines the mode to be used to combine MEKs specified by the **KEY** statement.

The following combine parameter string values are valid:

Table 56. Valid combine parameter string values

Value	Description
`XORHMACSHA512`	Combine MEKs with a round of XOR followed by a round of HMAC with SHA-512.
`XOR`	Combine MEKs with a round of XOR.

WRAP *WrapParamString*
specifies a string that defines the encryption algorithm and the wrapping mode to be used to wrap the FEK.

The following wrapping parameter string values are valid:

Table 57. Valid wrapping parameter string values.

Value	Description
`AES:KWRAP`	Use AES key wrap to wrap the FEK.
`AES:CBCIV`	Use AES in CBC-IV mode to wrap the FEK.

KEYS ('*Keyname***'[, '***Keyname***', ...])**
specifies one or more keys to be applied. Each *Keyname* is a unique identifier that combines the name of the key and the RKM server on which it resides. The format for *Keyname* is:

KeyId:*RkmId*

where

KeyId
An internal identifier that uniquely identifies the key inside the RKM. Valid characters for *KeyId* are the following: 'A' through 'Z'; 'a' through 'z'; '0' through '9'; and '-' (hyphen). The minimum length of *KeyId* is one character; the maximum length is 42 characters.

RkmId
The identifier of the `/var/mmfs/etc/RKM.conf` entry for the RKM that manages the key. An RKM ID must be unique within the cluster, must be 1-21 characters in length, and can contain only the characters a - z, A - Z, 0 - 9, or underscore (_). The first character cannot be a numeral.

Notes:

1. The maximum number of keys you can specify with the **ENCRYPTION IS** rule is eight.
2. The number of keys that can be used to encrypt a single file is permanently limited by the inode size of the file system.
3. You cannot specify the same key more than once in a given **ENCRYPTION IS** rule. Also, do not specify keys with identical values in an **ENCRYPTION IS** rule. Specifying the same key or identically-valued keys could result in a security breach for your data.

SET ENCRYPTION

The **SET ENCRYPTION** rule is similar to the **SET POOL** rule. If more than one such rule is present, all **SET ENCRYPTION** rules are considered and the FEK is wrapped once for each of the rules that apply (up to the maximum of eight). As mentioned in "Encryption keys" on page 367, if an FEK is wrapped multiple times, only one of the wrapped-FEK instances needs to be unwrapped for the file to be accessed.

If no **SET ENCRYPTION** rule is applicable at create time, the file is not encrypted.
The syntax of the **SET ENCRYPTION** rule is:

```
RULE 'RuleName' SET ENCRYPTION 'EncryptionSpecificationName'[, 'EncryptionSpecificationName',...]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [WHERE SqlExpression]
```

where:

EncryptionSpecificationName
is the name of a specification defined by an **ENCRYPTION IS** rule.

To stop traversing policy rules at a certain point and encrypt using only those rules that have matched up to that point, use the **SET ENCRYPTION EXCLUDE** rule:

```
RULE ['RuleName']  SET ENCRYPTION EXCLUDE
     [FOR FILESET ('FilesetName'[,'FilesetName']...)]
     [WHERE SqlExpression]
```

Note: Encryption policies do not support the **ACTION** clause.

Default encryption parameters

To simplify policy management, GPFS accepts the special default value `'DEFAULTNISTSP800131A'` as the **ALGO** parameter string.

For example, this policy statement:

```
RULE 'somerule' ENCRYPTION 'somename' IS
    ALGO 'DEFAULTNISTSP800131A'
    KEYS('KEY-2f1f7700-de74-4e55-a9be-bee49c5b3af8:RKMKMIP3')
```

corresponds to this:

```
RULE 'somerule' ENCRYPTION 'somename' IS
    ALGO 'AES:256:XTS:FEK:HMACSHA512'
    COMBINE 'XORHMACSHA512'
    WRAP 'AES:KWRAP'
    KEYS('KEY-2f1f7700-de74-4e55-a9be-bee49c5b3af8:RKMKMIP3')
```

Note: When this special **ALGO** default value is set as the **ALGO** *EncParamString*, neither **COMBINE** nor **WRAP** should be specified.

Example of an encryption policy

This is an example of an encryption policy:

```
RULE 'myEncRule1' ENCRYPTION 'E1' IS
         ALGO 'DEFAULTNISTSP800131A'
         KEYS('1:RKM_1', '2:RKM_2')

RULE 'myEncRule2' ENCRYPTION 'E2' IS
         ALGO 'AES:256:XTS:FEK:HMACSHA512'
         COMBINE 'XOR'
         WRAP 'AES:KWRAP'
         KEYS('3:RKM_1')

RULE 'myEncRule3' ENCRYPTION 'E3' IS
         ALGO 'AES:128:CBC:FEK:HMACSHA512'
         COMBINE 'XORHMACSHA512'
         WRAP 'AES:CBCIV'
         KEYS('4:RKM_2')

RULE 'Do not encrypt files with extension enc4'
         SET ENCRYPTION EXCLUDE
         FOR FILESET('fs1')
         WHERE NAME LIKE '%.enc4'

RULE 'Encrypt files with extension enc1 with rule E1'
         SET ENCRYPTION 'E1'
         FOR FILESET('fs1')
         WHERE NAME LIKE '%.enc1'

RULE 'Encrypt files with extension enc2 with rule E2'
         SET ENCRYPTION 'E2'
         FOR FILESET('fs1')
         WHERE NAME LIKE '%.enc2'

RULE 'Encrypt files with extension enc* with rule E3'
         SET ENCRYPTION 'E3'
         FOR FILESET('fs1')
         WHERE NAME LIKE '%.enc%'
```

Note:

In this example encryption policy:

- All files in fileset `fs1` are treated as follows:
  - If the extension is equal to `enc4`, the file is not encrypted. This happens because the `ENCRYPTION EXCLUDE` rule is matched first, stopping the traversal of the remaining rules before any additional matches can be made.
  - If the extension is equal to `enc1`, the file is encrypted with a 256-bit FEK, using AES in XTS mode; the FEK is preprocessed with HMAC with SHA-512, and the FEK is then wrapped twice:
    - once with AES key wrap, with keys `1:RKM_1` and `2:RKM_2` combined via one round of XOR followed by one round of HMAC with SHA-512
    - once with AES in CBC-IV mode using key `4:RKM_2`

    This happens because both rules `E1` and `E3` apply, since extension `enc1` matches both `%.enc1` and `%.enc%`. Note that the encryption algorithms specified by rule `E1`, which grant a stronger security than those of rule `E3`, are chosen and applied.
  - If the extension is equal to `enc2`, the file is encrypted with a 256-bit FEK, using AES in XTS mode; the FEK is preprocessed with HMAC with SHA-512; and the FEK is then wrapped twice:

  - once with AES key wrap using key `3:RKM_1`
  - once with AES in CBC-IV mode using key `4:RKM_2`

  This happens because both rules `E2` and `E3` apply, since extension `enc2` matches both `%.enc2` and `%.enc%`.
- If the extension is equal to `enc3`, the file is encrypted with a 128-bit FEK, using AES in CBC mode; the FEK is preprocessed with HMAC with SHA-512; and the FEK is then wrapped once with AES in CBC-IV mode using key `4:RKM_2`.

  This happens because only rule `E3` applies, since extension `enc3` only matches `%.enc%`.

- A GPFS node with access to both keys `1:RKM_1` and `2:RKM_2` or to key `4:RKM_2` can access a file with extension `enc1`.
- A GPFS node with access to key `3:RKM_1` or to key `4:RKM_2` can access a file with extension `enc2`.
- A GPFS node with access to key `4:RKM_2` can access a file with extension `enc3`.
- No key is required to access a file with extension `enc4`.
- A file with extension `enc1` is securely deleted when either key `1:RKM_1` or `2:RKM_2`, and key `4:RKM_2` are destroyed in their respective RKMs (and their cached copies have been flushed).
- A file with extension `enc2` is securely deleted when key `3:RKM_1` and key `4:RKM_2` are destroyed in their respective RKMs (and their cached copies have been flushed).
- A file with extension `enc3` is securely deleted when key `4:RKM_2` is destroyed in its respective RKM (and its cached copies have been flushed).
- Once created, a file may not be encrypted with more MEKs, only with different MEKs using the **REWRAP** rule.

Rewrapping policies

Rewrapping policies are used to change the way a set of FEKs is encrypted; that is, to change the set of MEKs that wrap the FEKs of those files. Rewrapping applies only to files that are already encrypted, and the rewrapping operation acts only on the `gpfs.Encryption` EA of the files. Rewrapping is done by using the **mmapplypolicy** command to apply a set of policy rules containing one or more `CHANGE ENCRYPTION KEYS` rules. These rules have the form:

```
RULE 'ruleName' CHANGE ENCRYPTION KEYS FROM 'Keyname_1' to 'Keyname_2'
[FROM POOL 'poolName']
  [FOR FILESET(...)]
  [SHOW(...)]
  [WHERE ... ]
```

where:

- *Keyname_1* is the unique identifier of the MEK to be replaced. (See "Encryption policy rules" on page 368 for *Keyname* format.)
- *Keyname_2* is the unique identifier of the new MEK, which will replace the old MEK identified by *Keyname_1*.
- The `FOR FILESET` and `WHERE` clauses narrow down the set of affected files.

Both *Keyname_1* and *Keyname_2* are listed, and only the files that currently use *Keyname_1* will have their FEKs rewrapped with *Keyname_2*. Files that do not currently use *Keyname_1* are not affected by the operation.

Notes:
1. Only the *first* matching **CHANGE ENCRYPTION KEYS** rule will be applied to each file. The rule will rewrap each wrapped version of the FEK that was encrypted with the MEK in the **CHANGE ENCRYPTION KEYS** rule.
2. The same MEK cannot be used more than once in a particular wrapping of the FEK.

Encryption setup

Encryption has the following setup requirements:

IBM Spectrum Scale software
GPFS Advanced Edition V4.1 or later is required for encryption.

IBM Spectrum Scale node setup
The following setup is required:

- Set up an operating system image with a running version of **mmfsd** deployed on a set of nodes.
- Set up an IBM Spectrum Scale cluster that is properly configured and is up and running.
- The file system must be GPFS V4.1 or later. The following setup is required:
  - Enable fast external attributes. This option is the default for newly created file systems. However, if the file system was migrated from an earlier level, it might be necessary to issue the **mmmigratefs** *FsName* **--fastea** command.
  - Set the inode size to 4 KB. This size is the recommended minimum size. The 4 KB inode size is recommended to accommodate the `gpfs.Encryption` extended attribute that is assigned to each encrypted file at file creation time. This extended attribute contains one or more wrapped FEKs, so it can potentially grow large. (See "Encryption policies" on page 368.)
- `gpfs.gskit` and `gpfs.crypto` packages installed

RKM Setup
The following setup is required:

- RKM server. The only server that is supported is IBM Security Key Lifecycle Manager (ISKLM) v2.5.0.1 or later.

  Note: ISKLM has a complete implementation of the Key Management Interoperability Protocol (KMIP) standard of the Organization for the Advancement of Structured Information Standards (OASIS). IBM Spectrum Scale nodes use the KMIP protocol to retrieve keys from ISKLM servers.
- IBM Spectrum Scale nodes with direct network access to the RKM server

RKM Backends
Nodes in the cluster must do encryption if they do any of the following actions:

- Data access (reading and writing files)
- Administration tasks such as snapshot deletion, disk repair, or file system check

Therefore, these nodes must be able to contact one or more RKM backends to transparently access keys that are required by the encryption policies.

The configuration of each RKM backend must be described in a `/var/mmfs/etc/RKM.conf` file on each such node. The `/var/mmfs/etc/RKM.conf` file does not have to be identical. By controlling the contents of this file, the cluster administrator can control which client nodes have access to which keys. For example, the same RKM server can be given two different names in `/var/mmfs/etc/RKM.conf` stanzas. Then, the administrator can partition a set of MEKs hosted on a single RKM server into separate subsets of MEK. These subsets of MEK might belong to subsets of the nodes of the cluster.

/var/mmfs/etc/RKM.conf file

The `/var/mmfs/etc/RKM.conf` file consists of a series of stanzas, each of which uses the following syntax:

```
# RKM entry
RkmId {
   type = ISKLM                          # The RKM is an ISKLM
   kmipServerUri = tls://host:port       # TLS connection to host on port
   keyStore = /PathToKeyStoreFile          # The path to the PKCS#12 file containing server certificate
                                           #  and client public/private keypair
   passphrase = Password                   # Passphrase protecting the keystore file
   clientCertLabel = LabelName             # Label of the client key pair to be used among those in the
                                           #  keystore file
   tenantName = NameOfTenant               # Name of tenant, set in IBM Security Key Lifecycle Manager set-up
   [connectionTimeout = ConnectionTimeout]   # Connection timeout, in seconds (default 60 seconds)
   [connectionAttempts = ConnectionAttempts] # Number of connection attempts (default 3)
   [retrySleep = RetrySleepUsec]           # Retry sleep time, in microsecsonds
                                           #  (default 100000 = 0.1 seconds)
}
```

Notes:

1. Each entry in the `/var/mmfs/etc/RKM.conf` file starts with an *RkmId* string that identifies the RKM. (Later this string is combined with unique *KeyId* identifiers to specify individual keys in encryption policy rules. See "Encryption policy rules" on page 368.)
2. The following limits apply:
   - 0 < *ConnectionAttempts* <= 10
   - 0 < *RetrySleepUsec* <= 10000000 (in microseconds)
   - 0 < *ConnectionTimeout* <= 120 (in seconds)
3. The file size of the `/var/mmfs/etc/RKM.conf` file cannot exceed 1 MiB.
4. No limit is set on the number of `/var/mmfs/etc/RKM.conf` entries, if the file size does not exceed the file size limit of 1 MiB.
5. The contents of the `/var/mmfs/etc/RKM.conf` file and of the keystore file are security-sensitive. It is a good idea to store them as node-local files. Also, IBM Spectrum Scale loads the content of security-sensitive files only if they meet the following requirements:
   - They are regular files that are owned by the root user.
   - They are in the root group.
   - They are readable and writable by the owner only.
6. After the file system is configured with encryption policy rules, the file system is considered encrypted. From that point on, each node with access to that file system must have an `RKM.conf` file present. Otherwise, the file system might not be mounted or might become unmounted.

Identifying multiple RKM backends in a high-availability configuration

The ISKLM supports automated replication across multiple nodes for high-availability deployments. To identify multiple RKM backends in a high-availability configuration, specify any of the following optional parameters:

```
rkmname3 {
...
   kmipServerUri2 = tls://host:port   # TLS connection to clone number 1 to host on port
   kmipServerUri3 = tls://host:port   # TLS connection to clone number 2 to host on port
   kmipServerUri4 = tls://host:port   # TLS connection to clone number 3 to host on port
   kmipServerUri5 = tls://host:port   # TLS connection to clone number 4 to host on port
   kmipServerUri6 = tls://host:port   # TLS connection to clone number 5 to host on port
...
}
```

If at least one backup is configured, whenever key retrieval from the master fails, IBM Spectrum Scale looks in each backup until it finds the MEK. The addition of the URIs for the clone servers is the only required change within IBM Spectrum Scale. All other configuration parameters (certificates, keys, node, and tenant information) do not need to change, because they are also part of the set of information that is replicated. The administrator is responsible for creating and maintaining any backups.

Additionally, setting up ISKLM key server clones can help gain some performance advantage by distributing MEK retrieval requests across the different clones in a round-robin fashion. To achieve this

| result, the administrator must specify different orderings of the server endpoints on different IBM
| Spectrum Scale nodes in the `/var/mmfs/etc/RKM.conf` file.

| For example, if two cloned ISKLM servers are available (such as `tls://keysrv.ibm.com:5696` and
| `tls://keysrv_backup.ibm.com:5696`), half of the nodes in the cluster can have the following content in
| `/var/mmfs/etc/RKM.conf`:

```
| ...
|    kmipServerUri  = tls://keysrv.ibm.com:5696
|    kmipServerUri2 = tls://keysrv_backup.ibm.com:5696
| ...
```

| The other half can use the following content:

```
| ...
|    kmipServerUri  = tls://keysrv_backup.ibm.com:5696
|    kmipServerUri2 = tls://keysrv.ibm.com:5696
| ...
```

|

Establishing an encryption-enabled environment

Establishing an encryption-ready environment requires the sequence of activities summarized here. This summary covers a basic setup with a single encrypted fileset.

Considerations on FIPS compliance

To retain compliance to FIPS 140-2, follow these steps:

- Ensure that the `fips` configuration parameter is turned to **on** on the ISKLM *before* you generate server certificates and MEKs. For more information, see the ISKLM installation guide.
- Ensure that GPFS is running with the **FIPS1402mode** configuration variable set to **yes** *before* you create the key store file.

Note: In IBM Spectrum Scale V4.2 and earlier, in a Power® 8, little-endian environment, the setting **FIPS1402mode**=**no** is required for the following operations:

- File encryption
- Secure communications between nodes. For more information, see the following descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:
  - **-l** *CipherList* parameter of the **mmauth** command
  - **cipherList** parameter of the **mmchconfig** command
- CCR enablement. For more information, see the following descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:
  - **--ccr-enable** parameter of the **mmchcluster** command
  - **--ccr-enable** parameter of the **mmcrcluster** command.

Initial setup by administrator

During initial setup, perform the following steps:

1. **Install IBM Security Key Lifecycle Manager (ISKLM) version 2.5.0.1 or later.**

   For information about installing ISKLM, see the *IBM Security Key Lifecycle Manager: Installation and Configuration Guide (SC27-5335)*.

   From the main page of the ISKLM web GUI, select **Configuration** and then **Key Serving Parameters** from the selector on the right. Then make sure that the check box next to **Keep pending client device communication certificates** is selected.
2. When operating in NIST SP 800-131A mode (the configuration variable **nistCompliance** is set to SP800-131A), configure the ISKLM key server to operate with NIST-Compliant ciphers and protocols.

- For Linux systems, the `/opt/IBM/WebSphere/AppServer/products/sklm/config/SKLMConfig.poperties` file located on the server must include this line: **TransportListener.ssl.protocols=TLSv1.2**
- For Windows systems, the file is located in `Drive:\Program Files\x86\IBM\ WebsSphere\AppServer\products\sklm\config\sklm\config\SKLMConfig.properties`

If FIPS compliance is required, enable it in ISKLM by setting the **fips=on** in `SKLMConfig.properties`

3. **Configure an SSL/KMIP server certificate on ISKLM.**

   Skip this step if you already configured the SSL/KMIP server certificate. If not, follow these steps:

   a. Click **Configuration** in the menu bar at the top, and select SSL/KMIP from the menu on the left.

   b. To create a new self-signed certificate, click the **Create self-signed certificate** radio button.

   | **Important:** ISKLM version 2.6.0.0 has a known issue in which setting the property `TransportListener.ssl.ciphersuites` causes the key server to create server certificates that are signed using SHA1withRSA. To create server certificates that are signed using SHA256withRSA, you must set the following properties while the key server is running:

   ```
   print AdminTask.tklmConfigUpdateEntry('[-name autoScaleSignatureHash -value true]')
   print AdminTask.tklmConfigUpdateEntry('[-name requireSHA2Signatures -value true]')
   ```

   | But restarting the key server causes ISKLM to revert to creating server certificates that are signed using SHA1withRSA.

   c. Enter the appropriate details for the certificate to be created, and click **OK**

   Important: To verify that the server can operate with the new certificate, you must restart the server.

4. **Export the SSL/KMIP server certificate from ISKLM.**

   This step is required to obtain a trusted copy of the server certificate.

   a. Identify the certificate label for the SSL/KMIP server certificate currently in use.

   b. Click **Advanced Configuration** from the menu bar at the top, and select **Server Certificates** from the dropdown menu.

   c. Select the certificate identified as being "In Use," click **Modify**, and make note of the certificate label.

   d. From an administrator console on the ISKLM server, change the directory to the `bin` directory in `WAS_HOME`:

   - Windows systems:

     ```
     drive:\Program Files (x86)\IBM\WebSphere\AppServer\bin
     ```

   - Other systems such as Linux:

     ```
     /opt/IBM/WebSphere/AppServer/bin
     ```

   e. Type the following:

   - On Windows systems:

     ```
     wsadmin -username SKLMAdmin -password mypwd -lang jython
     ```

   - On other systems such as AIX or Linux:

     ```
     ./wsadmin.sh -username SKLMAdmin -password mypwd -lang jython
     ```

   f. Identify the correct UUID of the certificate:

   1) First issue the following **print** command, replacing *YOUR_LABEL* with the certificate label obtained in the previous step:

      ```
      print AdminTask.tklmCertList('[-alias YOUR_LABEL]')
      ```

      The system will respond with output similar to the following:

```
CTGKM0001I Command succeeded.
uuid = CERTIFICATE-7005029a-831d-405f-af30-4bf0177909de
alias = server
key store name = defaultKeyStore
key state = ACTIVE
issuer name = CN=server
subject name = CN=server
creation date = 13/03/2014 16:27:13 Eastern Daylight Time
expiration date = 09/03/2015 07:12:30 Eastern Daylight Time
serial number = 1394363550
```

2) Reissue the **print** command, this time specifying the UUID identified during the previous step:

```
print AdminTask.tklmCertExport('[-uuid
CERTIFICATE-7005029a-831d-405f-af30-4bf0177909de -format base64 -fileName
/root/srvcert]')
```

The certificate will be exported in the file specified after `-fileName` (in this example, `/root/srvcert`).

g. Copy the file into a temporary directory on the GPFS node you wish to configure for encryption (for example, to `/tmp/srvcert`).

5. **Create a new GPFS device group.**
   a. From the main page of the ISKLM web GUI, select **Advanced Configuration** and then **Device Group** from the menu bar at the top.
   b. On the next page, click the **Create** button.
   c. In the **Create Device Group** window, select the **GPFS** device family and enter an appropriate name (for example, "GPFS_TENANT1"); make note of this name, as it will be required later on.
   d. Once the device group has been created, the system will prompt you to add devices and keys. However, the next steps in this procedure will generate keys explicitly, so now just click **Close** to return to the main page.
   e. On the main page, select the GPFS tenant device group, click **Go to...**, and select **Manage keys and devices** from the dropdown menu.
   f. On the next screen, click **Add** and select **Key** from the dropdown menu.

   The system prompts you to specify the following:
   - the number of keys to be created
   - the three-letter prefix to be added to their name

     The name is the ISKLM internal name and is not used for GPFS encryption.

   Make note of the key UUID (in the example, `KEY-326a1906-be46-4983-a63e-29f005fb3a15`), as it will be required later on.
   g. Enable the option **Hold new certificate requests pending my approval** by selecting it from the dropdown menu in the lower part of the page.
6. **Configure the RKM back end on GPFS.**
   a. Create a client key store that contains the server certificate, a new key pair, and a client certificate:
      1) On the GPFS node that you want to configure with encryption, in the administrator console, create the subdirectory `/var/mmfs/etc/RKMcerts` to contain the key store.
      2) Run the **mmauth** command with the **gencert** option to create the key store:

```
mmauth gencert --cname cname_test --label a_label --cert /tmp/srvcert --out
/var/mmfs/etc/RKMcerts/ISKLM.p12 --pwd-file password_file
```

where:

`cname_test` is an arbitrary name for the GPFS node or the tenant that is performing encryption.

`a_label` is the name of the client certificate in the RKM key store.

`/tmp/srvcert` is the path and file name of the RKM key store.

`/var/mmfs/etc/RKMcerts/client.p12` is the path and file name of the new client key store.

`pwd-file` is a file that contains a password to protect the secret key material in the key store file.

If you do not use --pwd-file option, the command prompts for the passphrase.

The new key store is created: `/var/mmfs/etc/RKMcerts/ISKLM.p12`.

Note: The new key store must be record locked when the GPFS daemon starts. If the key store files are stored on an NFS mount, it is possible for the encryption initialization process to hang, due to a bug that affects the way NFS handles record locking. If this happens, upgrade your version of NFS or store your key store file on a local file system. If an upgrade is not possible and no local file system is available, use a RAM drive to store the key store files.

b. Create a client certificate file.

   The RKM server does not trust unknown key stores in pkcs12 format. It accepts only client certificates that are built from the RKM server certificate. To create a client certificate of this type, you must generate a certificate from the new client key store and delete the server portion. Follow these steps:

   1) Generate a certificate from the new client key store. The following command generates the certificate file `ISKLM.crt` from the client key store `ISKLM.p12`:

      ```
      openssl pkcs12 -in ISKLM.p12 -nokeys -out ISKLM.crt
      ```

   2) The new certificate file contains two certificates, a server certificate and a client certificate. Delete the server certificate:

      a) Open the client certificate file with a text editor. The following block of text shows a sample of the contents. The ellipsis and square brackets [...] indicates text that is not displayed in this example:

         ```
         neymar:/var/mmfs/etc/RKMcerts # cat neymar.crt
         Bag Attributes
         friendlyName: server
         [...]
         issuer=/CN=germanies key
         -----BEGIN CERTIFICATE-----
         MIIC0TCCAbmgAwIBAgIGC8MTPXV6MA0GCSqGSIb3DQEBCwUAMBgxFjAUBgNVBAMT
         [...]
         -----END CERTIFICATE-----
         Bag Attributes
         friendlyName: neymar
         localKeyID: 03 82 01 01 00 2D 90 3C 69 F1 F2 0F 05 BA 75 A5 AB 46 3D 70 7B D9 46 24 03 5F 78
         A6 EF BD 09 67 74 39 78 E3 89 40 46 0A 09 17 01 2E CC 1F 1B E1 52
         [...]
         subject=/CN=neymar
         issuer=/CN=neymar
         -----BEGIN CERTIFICATE-----
         [...] wmCPCDMys+4SbXvbNAjKxsKRZk2LtnKV5c=
         -----END CERTIFICATE-----
         ```

      b) The server certificate is the first certificate in the file. To delete it, delete the block of text that begins with the first `Bag Attributes` line (the second line of the example in the previous step) and that ends with the first `-----END CERTIFICATE-----` line. The following example shows the contents of the file after the deletion:

         ```
         neymar:/var/mmfs/etc/RKMcerts # cat neymar.crt
         Bag Attributes
         friendlyName: neymar
         localKeyID: 03 82 01 01 00 2D 90 3C 69 F1 F2 0F 05 BA 75 A5 AB 46 3D 70 7B D9 46 24 03 5F 78
         A6 EF BD 09 67 74 39 78 E3 89 40 46 0A 09 17 01 2E CC 1F 1B E1 52
         [...]
         subject=/CN=neymar
         issuer=/CN=neymar
         -----BEGIN CERTIFICATE-----
         [...] wmCPCDMys+4SbXvbNAjKxsKRZk2LtnKV5c=
         -----END CERTIFICATE-----
         ```

Close the file.

c. Copy the edited client certificate file (ISKLM.crt) to the key server node.

d. Add the client certificate to the Device Group that you created in Step 5:

   1) In the ISKLM server GUI, on the **Welcome** tab, select the Device Group and click **Go to** > **Manage keys and certificates**.

   2) In the window that opens, click **Add certificate** and follow the steps to add the client certificate from the client certificate file (ISKLM.crt) to the device group.

7. Open an editor to create the /var/mmfs/etc/RKM.conf file, as follows:

```
ISKLM_srv {
type = ISKLM
kmipServerUri = tls://raclette.zurich.ibm.com:5696
keyStore = /var/mmfs/etc/RKMcerts/ISKLM.p12
passphrase = a_password
clientCertLabel = a_label
tenantName = GPFS_TENANT1
}
```

Choose an appropriate name for the RKM backend stanza (ISKLM_srv in the example) and ensure that the fields are filled in correctly:

type Must be set to ISKLM.

kmipServerUri
Contains the DNS name or IP address of the ISKLM server and the KMIP SSL port (you can find the latter in the main page of the ISKLM web GUI; 5696 is the default).

keyStore
Contains the path to the key store file created in the previous step.

passphrase and clientCertLabel
Contain the password and label specified in the command line upon creation of the key store.

tenantName
Contains the name of the device group created in ISKLM.

8. **Set up an encryption policy.**

a. Create a policy, instructing GPFS to encrypt all files in the file system with an FEK and wrap the FEK with the MEK created previously; for example:

```
RULE 'p1' SET POOL 'system' # one placement rule is required at all times

RULE 'Encrypt all files in filesystem with rule E1'
     SET ENCRYPTION 'E1'
     WHERE NAME LIKE '%'

RULE 'simpleEncRule' ENCRYPTION 'E1' IS
     ALGO 'DEFAULTNISTSP800131A'
     KEYS('KEY-326a1906-be46-4983-a63e-29f005fb3a15:ISKLM_srv')
```

b. Create the *Keyname* string in the policy statement as follows:

KeyID
The key ID copied from the ISKLM web GUI.

RkmID
The name of the RKM backend stanza in the /var/mmfs/etc/RKM.conf file.

c. After installing the rule with **mmchpolicy**, you need to import the client certificate into ISKLM. To do that, attempt the creation of an encrypted file to trigger a KMIP request, for example:

```
[root@baden ~]# touch /gpfs0/test
touch: cannot touch `/gpfs0/test': Permission denied
[root@baden ~]# tail -n 2 /var/adm/ras/mmfs.log.latest
Thu Mar 20 14:00:55.029 2014: [E] Unable to open encrypted file: inode 46088,
Fileset fs1, File System gpfs0.
Thu Mar 20 14:00:55.030 2014: [E] Error: key
'KEY-326a1906-be46-4983-a63e-29f005fb3a15:ISKLM_srv' could not be fetched (RKM
reported error -1004).
```

The request will initially fail because ISKLM does not yet trust the client certificate; however, the certificate will be placed in a list of pending certificates.

d. From the main page of the ISKLM web GUI, select **Pending client device communication certificates**.
e. Identify the client certificate and click **View**.
f. Carefully check that the certificate being imported matches the one created in the previous step, then click **Accept and Trust**
g. On the resulting screen, give a name to the certificate, then click **Accept and Trust**.
h. Attempt again the file creation that previously failed to verify that the setup is complete:

```
[root@baden ~]# touch /gpfs0/test
[root@baden ~]# mmlsattr -n gpfs.Encryption /gpfs0/test
file name: /gpfs0/test
gpfs.Encryption: "EAGC????f?????????????? ??????w?^??>???????????? ?L4??
_-???V}f???X????,?G?<sH??0?)??M?????)?KEY-326a1906-be46-4983-a63e-29f005fb3a15?
isklmsrv?)?KEY-6aaa3451-6a0c-4f2e-9f30-d443ff2ac7db?RKMKMIP3?"
EncPar 'AES:256:XTS:FEK:HMACSHA512'
        type: wrapped FEK WrpPar 'AES:KWRAP' CmbPar 'XORHMACSHA512'
               KEY-326a1906-be46-4983-a63e-29f005fb3a15:isklmsrv
```

All files created from this point on will be encrypted with an FEK, which in turn will be wrapped with the chosen MEK.

Security note: The contents of /var/mmfs/etc/RKM.conf and of the key store file (/var/mmfs/etc/RKMcerts/ISKLM.p12 in the example) are extremely security-sensitive. GPFS will only load the content of security-sensitive files if they are regular files owned by the root user and are in the root group, and if they are readable and writable by the owner only, for example:

```
-rw-------. 1 root root 2446 Mar 20 12:15 /var/mmfs/etc/RKM.conf
drw-------. 2 root root 4096 Mar 20 13:47 /var/mmfs/etc/RKMcerts
-rw-------. 1 root root 3988 Mar 20 13:47 /var/mmfs/etc/RKMcerts/ISKLM.p12
```

Also, make sure the passphrase is not leaked through other means (for example, the shell history). Finally, note that GPFS neither manages nor replicates these files, so make sure you take appropriate precautions to ensure that they are not lost or corrupted.

Enabling encryption on other nodes

If you want to replicate the same encryption configuration on another node, you can simply replicate the content of /var/mmfs/etc on the latter. You may also choose to have different encryption configurations on different nodes; to do so, follow the steps described previously. Note the following:

- The content of /var/mmfs/etc/RKM.conf may be different across nodes.
- Different nodes may have different key stores.
- Installing a new encryption policy removes the previous one; as a consequence, all policy statements for the file system must be collected into a single file that can then be installed in the file system.

The scenario described here implies that the ability to create, read or write files from a particular node depends on the content of /var/mmfs/etc/RKM.conf and on which key stores are deployed on that node. For example, within a given cluster, read/write operations may succeed on one node and fail on another due to the fact that the right key store or /var/mmfs/etc/RKM.conf entry, or both, are missing.

Secure deletion

Secure deletion refers to both erasing files from the file system and erasing the MEKs that wrapped the FEKs that were used to encrypt the files.

Securely deleting files in a fileset

After files have been removed from a fileset using standard file system operations (such as **unlink** and **rm**), the tenant administrator might decide to securely delete them. For example, suppose that until that point, the FEKs of all files in the fileset were encrypted with the MEK with key name `KEY-old:isklmsrv`. To cause the secure deletion of all removed files, the administrator must perform the following steps:

1. Create a new MEK and note its key name (in this example, `KEY-new:isklmsrv`).
2. Modify the appropriate encryption policy **KEYS** statement in the encryption policy to encrypt new files with the new MEK (for example, `KEY-new:isklmsrv`) instead of the old one (`KEY-old:isklmsrv`).
3. Create and apply a migration (rewrapping) policy (**CHANGE ENCRYPTION KEYS**) to scan all files, unwrap the wrapped FEK entries of files that have been wrapped with the old key (`KEY-old:isklmsrv`), and rewrap them with the new key (`KEY-new:isklmsrv`); this step ensures that the FEKs of existing files will be accessible in the future.
4. Remove the old key, `KEY-old:isklmsrv`. This step commits the secure deletion of all files that were previously unlinked (and whose FEKs had therefore not been rewrapped with the new MEK, `KEY-new:isklmsrv`).
5. On each node that has ever done I/O to a file encrypted with the old key (`KEY-old:isklmsrv`), run the following command:
   ```
   /usr/lpp/mmfs/bin/tsctl encKeyCachePurge 'KEY-old:isklmsrv'
   ```

From this point on, the new key will be used for encryption, which will be performed transparently to the application.

Note: The **mmdelfs** command will *not* perform any secure deletion of the files in the file system to be deleted. **mmdelfs** only removes all the structures for the specified file system. To securely delete files, you need to perform the following steps:

1. Identify all MEKs currently used to wrap the FEKs of files in the file system to be deleted. If this information is not available through other means, obtain it by doing the following:
   a. Invoke **mmlsattr -n gpfs.Encryption** on all files of the file system.
   b. Parse the resulting output to extract all the distinct key names of the MEKs that are used.

   Note: These are the possible ways that an MEK might be in use in a file system:
   a. The MEK is, or was at some point, specified in an encryption rule in the policy set on the file system.
   b. An FEK rewrap has been run, rewrapping an FEK with another MEK.
2. Determine whether the identified MEKs were used to wrap FEKs in other file systems.

   WARNING: If the same MEKs were used to wrap FEKs in other file systems, deleting those MEKs will result in irreparable data loss in the other file systems where those MEKs are used. Before deleting such MEKs from the key servers, you must create one or more new MEKs and rewrap the files in the other file systems.
3. After appropriately handling any MEKs that were used to wrap FEKs in other file systems (as explained in the warning), delete the identified MEKs from their RKMs.

Secure deletion and encryption key cache purging

The key servers that store the MEKs know how to manage and securely delete keys. After an MEK is gone, all files whose FEKs were encrypted with that MEK are no longer accessible. Even if the data blocks corresponding to the deleted files are retrieved, the contents of the file can no longer be reconstructed, since the data cannot be decrypted.

However, if the MEKs have been cached for performance reasons (so that they do not have to be fetched from the server each time a file is created or accessed), the MEKs must also be purged from the cache to complete the secure deletion.

You can use the following command to purge a given key from the key cache, or to clean the entire cache, of an individual node:

```
/usr/lpp/mmfs/bin/tsctl encKeyCachePurge {Key | all}
```

where:

Key
: is the key ID, specified with the *KeyId*:*RkmId* syntax.

all
: specifies that the entire key cache is to be cleaned.

The scope of this command is limited to the local node and must be run on all nodes that have accessed the MEKs you are purging in order to ensure secure deletion.

Encryption and FIPS compliance

The **FIPS1402mode** configuration variable controls whether IBM Spectrum Scale uses encryption software modules that are certified according to the Federal Information Processing Standard (FIPS) 140 Publication Series.

In FIPS 140-2 mode, the product uses one or more of the following FIPS 140-2-approved cryptographic providers. The certificates are listed on the NIST website:

- IBMJCEFIPS (certificate 376)
- IBMJSSEFIPS (certificate 409)
- IBM Crypto for C (ICC) (certificate 384) for cryptography

To change the value of **FIPS1402mode**, run the **mmchconfig** command:

```
mmchconfig FIPS1402mode=yes
```

The valid values are **yes** and **no**. The default is **no**.

Note: When the mode is **no**, Linux nodes use kernel encryption modules for direct I/O. If you then set the mode to **yes**, the direct I/O performance of Linux nodes is degraded. To resolve this problem, restart the GPFS daemon on the Linux nodes, so that the daemon can pick up the new setting.

Note: In IBM Spectrum Scale V4.2 and earlier, in a Power 8, little-endian environment, the setting **FIPS1402mode=no** is required for the following operations:

- File encryption
- Secure communications between nodes. For more information, see the following descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:
  - **-l** *CipherList* parameter of the **mmauth** command
  - **cipherList** parameter of the **mmchconfig** command

- CCR enablement. For more information, see the following descriptions in the *IBM Spectrum Scale: Administration and Programming Reference*:
  - **--ccr-enable** parameter of the **mmchcluster** command
  - **--ccr-enable** parameter of the **mmcrcluster** command.

Encryption and NIST compliance

Encryption always uses NIST-compliant mechanisms.

Encryption and backup/restore

GPFS will deliver all data to **mmbackup** and other external backup solutions in **cleartext** whether or not the data is encrypted in GPFS. Any backups that are taken **will not** preserve the encryption status or the encrypted content of the data. Files that are recreated upon restore will be considered for encryption status based on the policy in place on the file system at the time of the restore operation.

Encryption and snapshots

GPFS preserves the encryption status of files when they are copied into global or fileset snapshots. Global snapshot restore will restore files precisely as they are in the snapshot, including FEKs and MEKs. For details about how fileset snapshot restore functions, see the description of the **mmrestorefs** command in the *IBM Spectrum Scale: Administration and Programming Reference*.

Chapter 20. Managing certificates to secure communications between GUI web server and web browsers

The IBM Spectrum Scale system supports self-signed and trusted certificates that are provided by a certificate authority (CA) to secure communications between the system and web browser.

During system setup, an initial self-signed certificate is created to use for secure connections between the GUI web servers and web browsers. Based on the security requirements for your system, you can create either a new self-signed certificate or install a signed certificate that is created by certify authority. Self-signed certificates can generate web browser security warnings and might not comply with organizational security guidelines. The self-signed certificates are stored in the Liberty profile SSL keystore, which is located at the `resources/security` directory in the server. You can find this directory in the following path: `/opt/ibm/wlp/usr/servers/gpfsgui/resources/security`.

The trusted certificates are created by a third-party certificate authority. These certificate authorities ensure that certificates have the required security level for an organization based on purchase agreements. Trusted certificates usually have higher security controls for encryption of data and do not cause browser security warnings. Trusted certificates are also stored in the Liberty profile SSL keystore.

Major web browsers trust the CA-certified certificates by default and therefore they can confirm that the certificate received by the GUI server can be trusted. You can either buy a signed certificate from a trusted third-party authority or create you own certificate and get it certified. You can use both self-signed and trusted certificates. However, using a trusted is the preferred way because the browser trusts this certificate automatically without any manual interventions.

Obtaining and importing a signed-certificate from a trusted certificate authority

You need to perform the following steps to obtain and import a signed-certificate from a trusted certificate authority:

1. Generate a private key by issuing the following command:
   ```
   openssl genrsa -out <nameOfYourKey>.key 2048
   ```
2. Generate the certificate request as shown in the following example:
   ```
   openssl req -new -key <nameOfYourKey>.key -out <nameOfYourKey>.csr
   ```
   The system prompts you to enter the following details:
   ```
   Country Name (2 letter code) [XX]:
   State or Province Name (full name) []:
   Locality Name (eg, city) [Default City]:
   Organization Name (eg, company) [Default Company Ltd]:
   Organizational Unit Name (eg, section) []:
   Common Name (eg, your name or your server's hostname) []:
   Email Address []:
   Please enter the following 'extra' attributes to be sent with your certificate request
   A challenge password []:
   An optional company name []:
   ```
3. Send the certificate request to a trusted certificate authority to get a certificate file.
4. Create a PKCS12 store containing the certificate as shown in the following example:
   ```
   openssl pkcs12 -export -in <yourCertificateFile> -inkey <nameOfYourKey>.key
   <nameOfYourPKCS12File>.p12
   ```
   The system prompts to set the export password as shown in the following example:
   ```
   Enter export Password: <yourPasswrod>
   Verifying - Enter export Password: <yourPasswrod>
   ```

5. Generate a Java keystore file (.jks) by using the keytool. It is stored in the following directory: `/opt/ibm/wlp/java/jre/bin`. Ensure that you set the paths to the Java keystore file properly in the system. Issue the following commands to generate a Java keystore file.

   ```
   <PathToKeytool>/keytool -importkeystore -srckeystore
   <NameOfYourPKCS12File>.p12 -destkeystore
   <NameOfYourJKSFile>.jks -srcstoretype pkcs12
   ```

   The system prompts you to enter the destination keystore password. You need to use the same password that you used while creating the PKCS12 store.

   ```
   Enter destination keystore password: <yourPassword>
   Re-enter new password: <yourPassword>
   Enter source keystore password: <yourPassword>
   ```

6. Copy the new Java keystore file to the directory named `security`, which is stored in the GUI server. This directory is located at the following location in the GUI server: `/opt/ibm/wlp/usr/servers/gpfsgui/resources/security`. It is the default place where keystore files are stored.

   ```
   cp <NameOfYourJKSFile>.jks <pathToSecurityDir>
   ```

7. You need to define the password to access the Java Keystone file in the `server.xml` file, which is stored in the GUI server. If you want to encode your password so that it does not get stored in plain text, use a security utility, which is stored in the following directory: `/opt/ibm/wlp/bin`. The supported encodings are XOR and AES.

   ```
   <PathToSecurityUtility>/securityUtility encode --encoding=<xor or aes> <yourPassword>
   ```

8. Edit keystore entry of the `server.xml` file for the GUI server to enable the new key. The `server.xml` file is located at the following location: `/opt/ibm/wlp/usr/servers/gpfsgui`. The following entry is on one line:

   ```
   <keyStore id="defaultKeyStore" password="<yourPassword> or <yourEncodedPassword>"
    location="<nameOfYourJKSFile>.jks" />
   ```

9. Restart the IBM Spectrum Scale management GUI by issuing the following commands:

   ```
   For RHEL7 or SLES12: systemctl restart gpfsgui
   For RHEL6: service gpfsgui restart
   ```

Chapter 21. Securing protocol data

The data cannot be secured only by authenticating and authorizing the users to access the data. You also need to ensure that the communication channel that is used to raise authentication requests and data transfer is secured. The security features associated with the protocols that you use to store and access data also help to provide data in transit security for the protocol data.

The secured data access by clients through protocols is achieved through the following two steps:

1. Establishing secured connection between the IBM Spectrum Scale system and the authentication server.

   When the client raises an authentication request to access the data, the IBM Spectrum Scale system interacts with the external authentication servers like Active Directory or LDAP based on the authentication configuration. You can configure the security services like TLS and Kerberos with the external authentication server to secure the communication channel between the IBM Spectrum Scale system and the external authentication server.
2. Securing the data transfer.

   The actual data access wherein the data transfer is made secured with the security features that are available with the protocol that you use to access the data.

The following diagram depicts the data in transit security implementation in the IBM Spectrum Scale system.

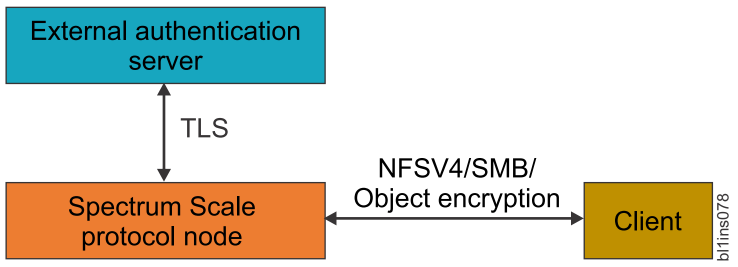

External authentication server
TLS
Spectrum Scale protocol node
NFSV4/SMB/ Object encryption
Client
bl1ins078

Figure 16. Implementation of data in transit security for protocol data

Secured connection between the IBM Spectrum Scale system and the authentication server

You can configure the following authentication servers to configure file and object access:

- Microsoft Active Directory (AD)
- Lightweight Directory Access Protocol (LDAP)
- Keystone

AD and LDAP can be used as the authentication server for both file and object access. Configuring the Keystone server is a mandatory requirement for the object access to function. The keystone needs to interact with the authentication server to resolve the authentication requests. You can configure either an internal or external keystone server for object access. The following table lists the security features that are used to secure the corresponding authentication server.

Table 58. Security features that are used to secure authentication server.

Authentication server	Supported protocols	Security features
Active Directory	File and Object	Kerberos for file and TLS for object.

Table 58. Security features that are used to secure authentication server (continued).

Authentication server	Supported protocols	Security features
LDAP	File and Object	Both TLS and Kerberos for file and only TLS for object.
Keystone	Object	SSL certificate to enable HTTPS connection

Secured data transfer

The secured data transfer over the network is based on the security features available with the protocols that are used to access the data.

Secured SMB data transfer

SMB protocol version 3 and later has the following capabilities to provide tighter security for the data transfers:

1. Secured dialect negotiation
2. Improved signing
3. Secured transmission

The dialect negotiation is used to identify the highest level dialect both server and client can support. The system administrator can enable SMB encryption by using the `smb encrypt` setting at the export level. The following three modes are available for the secured SMB access:

- Automatic
- Mandatory
- Disabled

When the SMB services are enabled, the SMB encryption is enabled in the automatic mode by default.

Note: SMB supports per-export encryption, which allows the administrators to selectively enable or disable encryption per SMB export.

Secured NFS data transfer

The following security methods are used with NFSV4 protocol:

1. Enabling squashing

   Any file requests that are made by the root user on the client system is considered as a potential threat. By default, root user requests are treated as if it is made by the user nobody on the server. If you disable squashing, the root user on the client gets the same level of access to files on the system as the root user on the server. You can disable squashing if, for example, you want to run an administrative task on the client system that has the exported directories that are stored on it.
2. Using Kerberos

   Kerberos is a network authentication protocol that ensures secure communication over a network. You can use Kerberos instead of local UNIX UIDs and GIDs to authenticate users. Kerberos can operate in the following modes to provide improved security:

   - **Kerberos v5:** Authentication only
   - **Kerberos v5 with integrity:** Authentication and data integrity
   - **Kerberos v5 with privacy:** Authentication and encryption of data traffic between the client and the server. Most secure, but it might cause some performance issues because of the heavy processing required for encryption.
3. Enabling port security

You can enable or disable the port security in all communications between the client and the server. When port security is enabled, the system does not allow access to the requests that originate from ports where the port number is greater than the hardcoded threshold value of 1024.

Note: The NFS security features can be configured per NFS export by the system administrator based on the requirement.

Secured object data access

The IBM Spectrum Scale system provides access to the Object Storage with the help of OpenStack Keystone Identity Service. The Keystone server that is provided by IBM Spectrum Scale is recommended to be used only for IBM Spectrum Scale Object workload.

For secure communication between the clients and the IBM Spectrum Scale Object, the system administrator needs to configure HAProxy for SSL termination, traffic encryption, and load balancing of the requests to IBM Spectrum Scale Object. The HAProxy needs to be set up on an external system that is not a part of the IBM Spectrum Scale cluster. For more information on how to configure HAProxy, see the documentation of the corresponding Linux distribution that you selected.

Planning for protocol data security

It is recommended to adhere to the following best practices when you plan to set up data security:

- When Windows clients use SMB 3.0, always configure SMB export with `smb encrypt = mandatory`.
- Avoid clients who used SMB 2.1 from accessing SMB data.
- Always enforce to use UNC with NetBIOS name of the cluster to access SMB data.
- To ensure NFSV4 encryption, use an authentication method that is configured with Kerberos.

Configuring protocol data security

The data security features associated with protocols facilitate to configure a secured way for the clients to raise the data access request and to transfer data from the IBM Spectrum Scale system to the client system.

Enabling secured connection between the IBM Spectrum Scale system and authentication server

You need to secure the communication channel between the IBM Spectrum Scale system and authentication server to secure the authentication server and hence to prevent unauthorized access to data and other system resources.

Securing AD server

To secure the AD server that is used for file access, configure it with Kerberos and to secure AD used for object access, configure it with TLS.

In the AD-based authentication for file access, Kerberos is configured by default. The following steps provide an example on how to configure TLS with AD, while it is used for object access.

1. Ensure that the CA certificate for AD server is placed under `/tmp` directory with the name *ldap_cacert.pem*; specifically, on the protocol node where the command is run. Perform validation of CA cert availability with desired name at required location as shown in the following example:

```
# stat /tmp/ldap_cacert.pem
File: ⌂/tmp/ldap_cacert.pem⌂
Size: 2130 Blocks: 8 IO Block: 4096 regular file
Device: fd00h/64768d Inode: 103169903 Links: 1
Access: (0644/-rw-r--r--) Uid: ( 0/ root) Gid: ( 0/ root)
```

```
Context: unconfined_u:object_r:user_tmp_t:s0
Access: 2015-01-23 12:37:34.088837381 +0530
Modify: 2015-01-23 12:16:24.438837381 +0530
Change: 2015-01-23 12:16:24.438837381 +0530
```

2. To configure AD with TLS authentication for object access, issue the **`mmuserauth service create`** command:

   ```
   # mmuserauth service create --type ad --data-access-method object
   --user-name "cn=Administrator,cn=Users,dc=IBM,dc=local"
   --password "myPassword" --base-dn "dc=IBM,DC=local"
   --enable-server-tls --ks-dns-name myKeystoneDnsName
   --ks-admin-user admin --servers myADserver
   --user-id-attrib cn --user-name-attrib sAMAccountName
   --user-objectclass organizationalPerson --user-dn "cn=Users,dc=IBM,dc=local"
   --ks-swift-user swift --ks-swift-pwd myKWSwiftPassword
   Object configuration with LDAP (Active Directory) as identity
   backend is completed successfully.
   Object Authentication configuration completed successfully.
   ```

   Note: The value that you specify for `--servers` must match the value in the TLS certificate; otherwise the command fails.

3. To verify the authentication configuration, use the **`mmuserauth service list`** command as shown in the following example:

   ```
   # mmuserauth service list
   FILE access not configured
   PARAMETERS               VALUES
   -------------------------------------------------

   OBJECT access configuration: AD
   PARAMETERS               VALUES
   -------------------------------------------------
   ENABLE_ANONYMOUS_BIND    false
   ENABLE_SERVER_TLS        true
   ENABLE_KS_SSL            false
   USER_NAME                cn=Administrator,cn=Users,dc=IBM,dc=local
   SERVERS                  myADserver
   BASE_DN                  dc=IBM,DC=local
   USER_DN                  cn=users,dc=ibm,dc=local
   USER_OBJECTCLASS         organizationalPerson
   USER_NAME_ATTRIB         sAMAccountName
   USER_ID_ATTRIB           cn
   USER_MAIL_ATTRIB         mail
   USER_FILTER              none
   ENABLE_KS_CASIGNING      false
   KS_ADMIN_USER            admin
   ```

Securing LDAP server

To secure the LDAP server that is used for file access, configure it with TLS and Kerberos and to secure LDAP server that is used for object access, configure it with TLS.

Provides examples of how to configure LDAP with TLS and Kerberos to secure the LDAP server when it is used for file and object access.

1. To configure LDAP with TLS and Kerberos as the authentication method for file access, issue the **`mmuserauth service create`** command as shown in the following example:

   ```
   # mmuserauth service create --type ldap --data-access-method file
   --servers es-pune-host-01 --base-dn dc=example,dc=com
   --user-name cn=manager,dc=example,dc=com --password secret
   --netbios-name ess --enable-server-tls --enable-kerberos
   --kerberos-server es-pune-host-01 --kerberos-realm example.com
   ```

   The system displays the following output:

   ```
   File Authentication configuration completed successfully.
   ```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
ENABLE_SERVER_TLS        true
ENABLE_KERBEROS          true
USER_NAME                cn=manager,dc=example,dc=com
SERVERS                  es-pune-host-01
NETBIOS_NAME             ess
BASE_DN                  dc=example,dc=com
USER_DN                  none
GROUP_DN                 none
NETGROUP_DN              none
USER_OBJECTCLASS         posixAccount
GROUP_OBJECTCLASS        posixGroup
USER_NAME_ATTRIB         cn
USER_ID_ATTRIB           uid
KERBEROS_SERVER          es-pune-host-01
KERBEROS_REALM           example.com

OBJECT access not configured
PARAMETERS               VALUES
-------------------------------------------------
```

2. To configure LDAP with TLS as the authentication method for object access, issue the **mmuserauth service create** command as shown in the following example:

```
# mmuserauth service create --type ldap --data-access-method object
--user-name "cn=manager,dc=essldapdomain" --password "Passw0rd"
--base-dn dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com --enable-server-tls
--ks-dns-name c40bbc2xn3 --ks-admin-user mamdouh --servers 192.0.2.11
--user-dn "ou=People,dc=essldapdomain" --ks-swift-user swift
--ks-swift-pwd Passw0rd
```

The system displays the following output:

```
Object configuration with LDAP as identity backend is completed successfully.
Object Authentication configuration completed successfully.
```

To verify the authentication configuration, use the **mmuserauth service list** command as shown in the following example:

```
# mmuserauth service list
```

The system displays the following output:

```
FILE access not configured
PARAMETERS               VALUES
-------------------------------------------------

OBJECT access configuration : LDAP
PARAMETERS               VALUES
-------------------------------------------------
ENABLE_ANONYMOUS_BIND    false
ENABLE_SERVER_TLS        true
ENABLE_KS_SSL            false
USER_NAME                cn=manager,dc=essldapdomain
SERVERS                  192.0.2.11
BASE_DN                  dc=isst,dc=aus,dc=stglabs,dc=ibm,dc=com
USER_DN                  ou=people,dc=essldapdomain
USER_OBJECTCLASS         posixAccount
USER_NAME_ATTRIB         cn
```

```
USER_ID_ATTRIB          uid
USER_MAIL_ATTRIB        mail
USER_FILTER             none
ENABLE_KS_CASIGNING     false
KS_ADMIN_USER           mamdouh
```

Securing Keystone server

The Keystone server that is used by the IBM Spectrum Scale system supports SSL. The SSL certificate provides secure communication while resolving the authentication requests. When Keystone is configured with authentication servers such as LDAP or AD, the system can be configured to establish a secured communication between AD or LDAP and Keystone by using TLS encryption. For more information on configuring AD or LDAP-based authentication with TLS, see the **`mmuserauth service create`** command. The IBM Spectrum Scale for Object Storage can also be configured with an external Keystone server. If the external Keystone server contains SSL certificate in place, then the system administrator can configure secured communication with the IBM Spectrum Scale system by following some manual steps.

Provides an example on how to configure secured object access.

1. Ensure that the following certificates are placed at the following location in the IBM Spectrum Scale system:
   - `certfile = /etc/keystone/ssl/certs/signing_cert.pem`
   - `keyfile = /etc/keystone/ssl/private/signing_key.pem`
   - `ca_certs = /etc/keystone/ssl/certs/signing_cacert.pem`

   To provide customized signing keys, follow these steps:
   a. Put the following three keys in the `/tmp` folder:
      - `/tmp/signing_cacert.pem`
      - `/tmp/signing_cert.pem`
      - `/tmp/signing_key.pem`

   b. Specify the **--enable-ks-casigning** parameter when you run the **mmuserauth** command in Step 2.
2. Issue the **`mmuserauth service create`** command as shown in the following example to configure local authentication method for object.
   ```
   # mmuserauth service create --type local --data-access-method object
   --ks-dns-name c40bbc2xn3 --ks-admin-user admin --ks-admin-pwd Passw0rd
   --enable-ks-casigning
   /usr/lpp/mmfs/bin/mmcesobjcrbase: Configuring Keystone server in
   /gpfs/cesfs/ces/object/keystone
   Object configuration with local (Database) as identity backend
   is completed successfully.
   Object Authentication configuration completed successfully.
   ```
3. Issue the **`mmuserauth service list`** command to verify the authentication configuration.
   ```
   # mmuserauth service list
   FILE access not configured
   PARAMETERS              VALUES
   -------------------------------------------------

   OBJECT access configuration : LOCAL
   PARAMETERS              VALUES
   -------------------------------------------------
   ENABLE_KS_SSL           false
   ENABLE_KS_CASIGNING     true
   KS_ADMIN_USER           admin
   ```

Securing data transfer

The data in transit security is configured by using the security features that are available with the protocol that is used for data I/O.

Securing NFS data transfer

Securing the NFS data transfer over the network is achieved by using the Kerberos-based encryption that is available with NFSV4 protocol. You can use Kerberos to encrypt the data that is transferred over the network and also to secure the communication with the authentication server.

The following example shows how to enable data security to ensure secured NFS data transfer.

1. 1. Create a keytab file for protocol nodes in IBM Spectrum Scale cluster. To create keytab file, you need to create a principal `nfs/<node-fqdn>` for each protocol node. Issue the following commands on the system that hosts the KDC server. In the following example, the sample commands are submitted on the Linux system that hosts MIT KDC server:

   ```
   $ addprinc -randkey nfs/<protocol-node1-fqdn>
   $ addprinc -randkey nfs/<protocol-node2-fqdn>

   .....

   &mldr; $ addprinc -randkey nfs/<protocol-nodeN-fqdn>
   $ ktadd -k /tmp/krb5.keytab nfs/<protocol-node1-fqdn>
   $ ktadd -k /tmp/krb5.keytab nfs/<protocol-node2-fqdn>

   .....

   &mldr; $ ktadd -k /tmp/krb5.keytab nfs/<protocol-nodeN-fqdn>
   ```

2. Ensure that the keytab file that is created is placed under the `/tmp` directory as *krb5.keytab*; specifically, on the node where the IBM Spectrum Scale authentication commands are submitted. Perform validation of keytab file availability with the required name and location:

   ```
   # stat /tmp/krb5.keytab
   File: ⌂/tmp/krb5.keytab⌂
   Size: 502 Blocks: 8 IO Block: 4096 regular file
   Device: fd00h/64768d Inode: 103169898 Links: 1
   Access: (0600/-rw-------) Uid: ( 0/ root) Gid: ( 0/ root)
   Context: unconfined_u:object_r:user_tmp_t:s0
   Access: 2015-01-23 14:31:18.244837381 +0530
   Modify: 2015-01-23 12:45:05.475837381 +0530
   Change: 2015-01-23 12:45:05.476837381 +0530
   Birth: -
   ```

3. Issue the **`mmuserauth service create`** command on the IBM Spectrum Scale protocol node as shown in the following example:

   ```
   #  mmuserauth service create --data-access-method file --type ldap
   --servers 192.0.2.17 --base-dn dc=example,dc=com
   --user-name "cn=manager,dc=example,dc=com" --password secret --enable-kerberos
   --kerberos-server 192.0.2.17 --kerberos-realm example.com --netbios-name cktest
   File Authentication configuration completed successfully.
   ```

4. Issue the **`mmuserauth service list`** command to see the current authentication configuration as shown in the following example:

   ```
   # mmuserauth service list
   ILE access configuration : LDAP
   PARAMETERS               VALUES
   -------------------------------------------------
   ENABLE_ANONYMOUS_BIND    false
   ENABLE_SERVER_TLS        false
   ENABLE_KERBEROS          true
   USER_NAME                cn=manager,dc=example,dc=com
   SERVERS                  9.118.46.17
   NETBIOS_NAME             cktest
   BASE_DN                  dc=example,dc=com
   USER_DN                  none
   GROUP_DN                 none
   NETGROUP_DN              none
   USER_OBJECTCLASS         posixAccount
   GROUP_OBJECTCLASS        posixGroup
   ```

```
USER_NAME_ATTRIB        cn
USER_ID_ATTRIB          uid
KERBEROS_SERVER         9.118.46.17
KERBEROS_REALM          example.com

OBJECT access not configured
PARAMETERS              VALUES
-------------------------------------------------
```

5. Create Kerberos exports with krb5, krb5i, and krb5p security features on the IBM Spectrum Scale node.

```
# mmcrfileset gpfs0 krb5
Fileset krb5 created with id 2 root inode 47898.

# mmlinkfileset gpfs0 krb5 -J /ibm/gpfs0/krb5
Fileset krb5 linked at /ibm/gpfs0/krb5

# mmnfs export add /ibm/gpfs0/krb5 --client \*
\(ACCESS_TYPE=RW,SQUASH=no_root_squash,SECTYPE=krb5\)
The NFS export was created successfully.

# mmcrfileset gpfs0 krb5i
Fileset krb5i created with id 3 root inode 47900.

# mmlinkfileset gpfs0 krb5i -J /ibm/gpfs0/krb5i
Fileset krb5i linked at /ibm/gpfs0/krb5i

# mmnfs export add /ibm/gpfs0/krb5i --client \*
\(ACCESS_TYPE=RW,SQUASH=no_root_squash,SECTYPE=krb5i\)
The NFS export was created successfully.

# mmcrfileset gpfs0 krb5p
Fileset krb5p created with id 4 root inode 47895.

# mmlinkfileset gpfs0 krb5p -J /ibm/gpfs0/krb5p
Fileset krb5p linked at /ibm/gpfs0/krb5p

# mmnfs export add /ibm/gpfs0/krb5p --client \*
\(ACCESS_TYPE=RW,SQUASH=no_root_squash,SECTYPE=krb5p\)
The NFS export was created successfully.

# mmnfs export list
Path               Delegations    Clients
-------------------------------------------
/ibm/gpfs0/krb5    none           *
/ibm/gpfs0/krb5i   none           *
/ibm/gpfs0/krb5p   none           *
/ibm/gpfs0/nfsexp1 none           *
```

6. Issue the **mmnfs export list** command with `krb5` option to see the authentication only configuration.

```
# mmnfs export list -n /ibm/gpfs0/krb5

Path             Delegations  Clients Access_Type Protocols Transports Squash         Anonymous_uid Anonymous_gid SecType PrivilegedPort Export_id DefaultDelegation Manage_Gids NFS_Commit
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
/ibm/gpfs0/krb5    none         *       RW          3,4       TCP        NO_ROOT_SQUASH -2            -2            KRB5    false          2         none              false       false
```

7. Issue the **mmnfs export list** command with `krb5i` option to see the authentication and data integrity configuration.

```
# mmnfs export list -n /ibm/gpfs0/krb5i

Path              Delegations    Clients Access_Type Protocols Transports Squash         Anonymous_uid Anonymous_gid SecType PrivilegedPort Export_id DefaultDelegation Manage_Gids NFS_Commit
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
/ibm/gpfs0/krb5i  none           *       RW          3,4       TCP        NO_ROOT_SQUASH -2            -2            KRB5I   false          3         none              false       false
```

8. Issue the **mmnfs export list** command with `krb5p` option to see the authentication and privacy configuration.

```
# mmnfs export list -n /ibm/gpfs0/krb5p

Path              Delegations    Clients Access_Type Protocols Transports Squash         Anonymous_uid Anonymous_gid SecType PrivilegedPort Export_id DefaultDelegation Manage_Gids NFS_Commit
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
/ibm/gpfs0/krb5p  none           *       RW          3,4       TCP        NO_ROOT_SQUASH -2            -2            KRB5P   false          4         none              false       false
```

Securing SMB data transfer

Secured SMB data transfer can be enabled when you are using SMB3 and later.

You can either enable or disable encryption of the data in transit by using the **mmsmb export create** command as shown in the following example:

```
# mmsmb export create secured_export  /ibm/gpfs0/secured_export --option "smb encrypt=mandatory"
```

Secured object data transfer

For secure communication between the clients and the IBM Spectrum Scale Object, the system administrator needs to configure HAProxy for SSL termination, traffic encryption, and load balancing of the requests to IBM Spectrum Scale Object. The HAProxy needs to be set up on an external system that is not a part of the IBM Spectrum Scale cluster. For more information on how to configure HAProxy, see the documentation of the corresponding Linux distribution that you selected.

Data security limitations

The following are the protocol data security limitations:

- The file system encryption feature is not supported for securing file and object protocol data.
- The SMB encryption is available only on SMB3 and later. All the limitations that are identified by Microsoft also apply to SMB encryption. There are no SMB encryption limitations that are specific to IBM Spectrum Scale.
- Delegations cannot be used with NFS in the Kerberos environment, because they cause the NFSV4 server to crash. If you use NFS in the Kerberos environment, you should disable delegations.

Chapter 22. Highly-available write cache (HAWC)

Highly-available write cache (HAWC) reduces the latency of small write requests by initially hardening data in a non-volatile fast storage device prior to writing it back to the backend storage system.

Overview and benefits

Current disk drive systems are optimized for large streaming writes, but many workloads such as VMs and databases consist of many small write requests, which do not perform well with disk drive systems. To improve the performance of small writes, storage controllers buffer write requests in non-volatile memory before writing them to storage. This works well for some workloads, but the amount of NVRAM is typically quite small and can therefore not scale to large workloads.

The goal of HAWC is to improve the efficiency of small write requests by absorbing them in any nonvolatile fast storage device such as SSDs, Flash-backed DIMMs, or Flash DIMMs. Once the dirty data is hardened, GPFS can immediately respond to the application write request, greatly reducing write latency. GPFS can then flush the dirty data to the backend storage in the background.

By first buffering write requests, HAWC allows small writes to be gathered into larger chunks in the page pool before they are written back to storage. This has the potential to improve performance as well by increasing the average amount of data that GPFS writes back to disk at a time.

Further, when GPFS writes a data range smaller than a full block size to a block for the first time, the block must first be fully initialized. Without HAWC, GPFS does this by writing zeroes to the block at the time of the first write request. This increases the write latency since a small write request was converted into a large write request (for example, a 4K write request turns into a 1MB write request). With HAWC, this initialization can be delayed until after GPFS responds to the write request, or simply avoided altogether if the application subsequently writes the entire block.

To buffer the dirty data, HAWC hardens write data in the GPFS recovery log. This means that with HAWC, the recovery log must be stored on a fast storage device because if the storage device on which the recovery log resides is the same as the data device, HAWC will decrease performance by writing data twice to the same device. By hardening data in the recovery log, all incoming requests are transformed into sequential operations to the log. In addition, it is important to note that applications never read data from the recovery log, since all data that is hardened in the recovery log is always kept in the page pool. The dirty data in the log is only accessed during file system recovery due to improper shutdown of one or more mounted instances of a GPFS file system.

The maximum size of an individual write that can be placed in HAWC is currently limited to 64KB. This limit has been set for several reasons, including the following:

- The benefit of writing data to fast storage decreases as the request size increases.
- Fast storage is typically limited to a much smaller capacity than disk subsystems.
- Each GPFS recovery log is currently limited to 1GB. Every file system and client pair has a unique recovery log. This means that for each file system, the size of HAWC scales linearly with every additional GPFS client. For example, with 2 file systems and 10 clients, there would be 20 recovery logs used by HAWC to harden data.

Note that combining the use of HAWC with LROC allows GPFS to leverage fast storage on application reads and writes.

Applications that can benefit from HAWC

Typically, it is recommended to place GPFS metadata in a storage pool consisting of fast storage devices such as SSDs. Storing GPFS recovery logs in fast storage improves the performance of metadata-intensive workloads where the recovery log is heavily used and when GPFS is configured to replicate data.

With HAWC, storing the recovery log in fast storage has the added benefit that workloads that experience bursts of small and synchronous write requests (no matter if they are random or sequential) will also be hardened in the fast storage. Well-known applications that exhibit this type of write behavior include VMs, databases, and log generation.

Since the characteristics of fast storage vary greatly, users should evaluate their application workload with HAWC in their storage configuration to ensure a benefit is achieved. In general, however, speedups should be seen in any environment that either currently lacks fast storage or has very limited (and non-scalable) amounts of fast storage.

Restrictions and tuning recommendations for HAWC

When enabling HAWC, take the following restrictions and tuning recommendations into consideration:

Ping pong recovery log buffers

Ping pong recovery log buffers should *not* be enabled when the recovery log is stored on storage devices that can gracefully write data upon power failure. This includes SSDs, NVRAM, storage controllers, RAID controllers among others.

Ping pong buffers are only needed to avoid data corruption when the recovery log is stored directly on disk. They place log data in two separate locations on disk to avoid loss of that data if a sector becomes unavailable. Writing to two separate locations creates additional overhead that is exacerbated by HAWC due to the large amount of data it places in the recovery log.

In general, it is not recommended to use HAWC with any storage device that would require ping pong buffers to be enabled because it doubles the amount of data that must be written before GPFS can respond to a application write request.

To disable log buffers, run the following command:

```
mmchconfig logPingPongSector=no
```

Recovery log size

The size of the recovery log defaults to a very small value (less than 16MB), which is not sufficient space to buffer HAWC data. Therefore, it is recommended to increase the size of the log at least to 128M or larger (1GB maximum). However, the effect of a larger recovery log is that upon node failure, more data must be recovered into the storage system, which will increase the time it takes to recover. It is important to take this into account because applications will not be able to access data in the file system while recovery is running.

Encryption

Encrypted data is never stored in the recovery log, but instead follows the pre-GPFS 4.1.0.4 semantics for synchronous writes even if the HAWC threshold is set to a value greater than 0.

Small files and directory blocks

HAWC does not change the following:

- write behavior of small files when the data is placed in the inode itself
- write behavior of directory blocks or other metadata

Using HAWC

To use HAWC effectively, you must enable it with an appropriate threshold, ensure the correct storage of the GPFS recovery log, and properly perform other administrative tasks.

Enabling HAWC

To enable HAWC, set the write cache threshold for the file system to any value between 4K and 64K (specified in multiples of 4KB), as shown in the following examples:

```
mmchfs gpfsA --write-cache-threshold 32K
```

or

```
mmcrfs /gpfs/gpfsB /dev/gpfsB -F ./deiskdef2.txt -B1M --write-cache-threshold 32K
```

Once HAWC has been enabled, all synchronous write requests smaller or equal to the threshold will be placed in the recovery log, and a response will be immediately returned to the application. All synchronous write requests greater than the threshold will be written directly to the primary storage system as usual.

Methods to ensure the use of the recovery log on fast storage

There are two methods for storing the GPFS recovery log. The method to choose depends on the storage architecture in which HAWC will be used.

Currently, Method 1 is the easier and more common method because it uses a centralized fast storage device, already common in many existing GPFS storage architectures. Method 2 enables the recovery log to be stored within the GPFS clients themselves, which can reduce latency, but it requires fast storage within every GPFS client node.

In either case, proper setup is important to ensure that written data will survive node or disk failures and will improve the performance of small synchronous writes.

Method 1. Storing GPFS metadata on highly available storage appliances

In this method, GPFS data is stored on a centralized fast storage device such as any of the following:

- Storage controller with SSDs
- FlashSystems
- GSS with SSDs

This method is relatively simple to set up, as NSDs can be specified to contain only metadata upon their creation. As always, the metadata storage pool can be replicated for higher availability.

Method 2 . Storing the GPFS recovery log on GPFS client nodes

This method calls for the following setup:

- GPFS clients (or a subset of them) have fast storage devices installed in them.
- The GPFS recovery log is stored in the NVRAM on the GPFS clients (separate from GPFS metadata).

  To do this, specify the `system.log` pool for the NVRAM NSDs on the GPFS clients at file system creation time or when adding disks to the file system. When those NSDs are used to create the file system, the recovery log will be placed in the `system.log` pool (which is now defined to be on the GPFS clients) instead of in the system pool.
- Because GPFS clients are not highly available (meaning that if they failed, the data stored on them would become unavailable), the `system.log` pool must be replicated.

To specify a replication factor for the `system.log` pool, the **--log-replicas** option must be used. The **--log-replicas** option is intended to be used in conjunction with the `system.log` pool feature to place log files in a separate pool with replication different from other metadata in the system pool.

Note: Log replication can be changed dynamically with the **mmchfs** command followed by the **mmrestripefs** command. There is no separate option for maximum log replication; therefore, on a file system created with a single replica of the recovery log (**--log-replicas=1**), it is only possible to enable log replication if the file system was created with **-M 2** or **-M 3**.

Additional administrative tasks

Using HAWC may also require the following administrative tasks:

Adding or removing disks from the `system.log` pool

The **mmrestripefs -b** command will restripe log files. Therefore, if you have enough NVRAM for all the logs, but too much of the log data ended up on a small number of disks because of the way the file system was created, **mmrestripefs** can be used to rebalance the data across the nodes.

Handling node/disk failures in the `system.log` pool

If the log is replicated, this command will ensure that data is re-replicated automatically when a node/disk fails:

```
mmchconfig restripeOnDiskFailure=yes -i
```

You can use this command to fine-tune how quickly the data will be replicated after failure (to avoid re-replication in case of node reboots, for example):

```
mmchconfig metadataDiskWaitTimeForRecovery=seconds
```

The default value for **metadataDiskWaitTimeForRecovery** is 300 seconds.

Using HAWC with Existing File Systems

To enable HAWC with an existing file system (once the entire cluster has been brought up to the latest file system and configuration version using **mmchconfig release=LATEST**), follow one or more of the following guidelines.

- If the metadata pool is already stored on a fast storage device, simply increase the size of the recovery log to at least 128M prior to setting the HAWC threshold by running the following command:

  ```
  mmchfs Device -L LogFileSize
  ```

- If the metadata pool is not already on a fast storage device, it must first be migrated to a fast storage device (see the *IBM Spectrum Scale: Advanced Administration Guide* topic “Managing storage pools”). Once the file system metadata is stored on fast storage devices, increase the size of the recovery log (as explained previously) and set the HAWC threshold.

Chapter 23. Local read-only cache

Many applications benefit greatly from large local caches. Not only is the data available with very low latency, but the cache hit serves to reduce the load on the shared network and on the backend storage itself, thus benefiting all nodes, even those without large caches.

Local solid state disks (SSDs) provide an economical way to create very large caches. The SSD cache serves as an extension to the local buffer pool. As user data or metadata is evicted from the buffer pool in memory, it can stored in the local cache. A subsequent access will retrieve the data from the local cache, rather than from the home location. The data stored in the local cache, like data stored in memory, remains consistent. If a conflicting access occurs, the data is invalidated from all caches. In a like manner, if a node is restarted, all data stored in the cache is discarded.

In theory, any data or metadata may be stored in the local SSD cache, but the cache works best for small random reads where latency is a primary concern. Since the local cache typically offers less bandwidth than the backend storage, it may be unsuitable for large sequential reads. The configuration options provide controls over what is stored in the cache. The default settings are targeted at small random I/O.

The local read-only cache function is disabled by default. To enable it, the administrator must define an NSD to be used by local read-only cache. The local read-only cache disk is expected to be a solid state disk (SSD) accessible via SCSI. The device as defined as a standard NSD by **mmcrnsd**, but the **DiskUsage** is set to **localCache**. The NSD must have a primary server and is not allowed to have other servers. The primary server must be the node where the physical local read-only cache device is installed. The device is *not* exported to other nodes in the cluster. The storage pool and failure group defined for NSD are ignored and should be set to null. The **mmcrnsd** command creates a unique NSD ID and name for the device and updates sector 2 to indicate that it is an NSD.

The minimum size of a local read-only cache device is 4 GB. The local read-only cache requires memory equal to 1% of the capacity of the local read-only cache device.

Once the local read-only cache disk is defined, the daemon code at the primary server node is automatically told to do device discovery. The daemon will detect that **localCache** is defined for its use and will determine the mapping to the local device. The daemon then informs the local read-only cache code to begin using the device for caching. Currently, there is a limit of four **localCache** devices per node. Note that the daemon code does not need to be restarted to begin using the cache.

The local read-only cache device may be deleted by using the **mmdelnsd** command. Both **mmcrnsd** and **mmdelnsd** may be issued while the daemon is running with file systems mounted and online. The call to delete the NSD first informs the daemon that the device is being deleted, which removes it from the list of active local read-only cache devices. Any data cached on the device is immediately lost, but data cached on other local read-only cache devices is unaffected. Once the **mmdelnsd** command completes, the underlying SSD may be physically removed from the node.

The NSD name for the local read-only cache device may not be used in any other GPFS commands, such as **mmcrfs**, **mmadddisk**, **mmrpldisk**, **mmchdisk** or **mmchnsd**. The device will be shown by **mmlsnsd** as a **localCache**.

Chapter 24. Miscellaneous advanced administration topics

The following topics provide information about miscellaneous advanced administration tasks:

- "Changing IP addresses and host names"
- "Enabling a cluster for IPv6" on page 404
- "Using multiple token servers" on page 404
- "Exporting file system definitions between clusters" on page 405
- "GPFS port usage" on page 405

Changing IP addresses and host names

GPFS assumes that IP addresses and host names remain constant. In the rare event that such a change becomes necessary or is inadvertently introduced by reinstalling a node with a disk image from a different node for example, follow the steps in this topic.

If all of the nodes in the cluster are affected and all the conditions in Step 2 following are met:

1. Run the **mmshutdown -a** command to stop GPFS on all nodes.
2. Using the documented procedures for the operating system, add the new host names or IP addresses but do not remove the old ones yet. This can be achieved, for example, by creating temporary alias entries in **/etc/hosts**. Do not restart the nodes until the **mmchnode** command in Step 3 is executed successfully. If any of these conditions cannot be met, use the alternate procedure described in this section.
3. Update the nodes that the cluster uses as the administration interface node and the daemon interface node, if necessary. To update these values, run the **mmchnode** command with the **--admin-interface** and the **--daemon-interface** options.

   Note: You cannot specify the **--daemon-interface** option for a quorum node if CCR is enabled. Temporarily change the node to a nonquorum node. Then run the **mmchnode** command with the **--daemon-interface** option against the nonquorum node. Finally, change the node back into a quorum node.
4. If the IP addresses over which the subnet attribute is defined are changed, you must update your configuration by running the **mmchconfig** command with the **subnets** attribute.
5. Start GPFS on all nodes with **mmstartup -a**.
6. Remove the unneeded old host names and IP addresses.

If only a subset of the nodes are affected, it may be easier to make the changes using these steps:

1. Before any of the host names or IP addresses are changed:
   - Use the **mmshutdown** command to stop GPFS on all affected nodes.
   - If the host names or IP addresses of the primary or secondary GPFS cluster configuration server nodes must change, use the **mmchcluster** command to specify another node to serve as the primary or secondary GPFS cluster configuration server.
   - If the host names or IP addresses of an NSD server node must change, temporarily remove the node from being a server with the **mmchnsd** command. Then, after the node has been added back to the cluster, use the **mmchnsd** command to change the NSDs to their original configuration. Use the **mmlsnsd** command to obtain the NSD server node names.
   - Use the **mmdelnode** command to delete all affected nodes from the GPFS cluster.
2. Change the node names and IP addresses using the documented procedures for the operating system.

3. If the IP addresses over which the subnet attribute is defined are changed, you need to update your configuration by using the **mmchconfig** command with the **subnets** attribute.
4. Issue the **mmaddnode** command to restore the nodes to the GPFS cluster.
5. If necessary, use the **mmchcluster** and **mmchnsd** commands to restore the original configuration and the NSD servers.

Enabling a cluster for IPv6

For newly created clusters, if any of the specified node interfaces on the **mmcrcluster** command resolves to an IPv6 address, the cluster is automatically enabled for IPv6. For existing IPv4-based clusters, follow one of the procedures that described in this section.

If you are performing the procedure during a scheduled maintenance window and GPFS can be shut down on all of the nodes in the cluster, run the command:

```
mmchconfig enableIPv6=yes
```

After the command finishes successfully, you can start adding new nodes with IPv6 addresses.

If it is not possible to shut down GPFS on all of the nodes at the same time, run the command:

```
mmchconfig enableIPv6=prepare
```

The next step is to restart GPFS on each of the nodes so that they can pick up the new configuration setting. This can be done one node at a time when it is convenient. To verify that a particular node has been refreshed, run:

```
mmdiag --config | grep enableIPv6
```

The reported value should be 1.

Once all of the nodes have been recycled in this manner, run the command:

```
mmchconfig enableIPv6=commit
```

This command will only succeed when all GPFS daemons have been refreshed. Once this operation succeeds, you can start adding new nodes with IPv6 addresses.

To convert an existing node from an IPv4 to an IPv6 interface, use one of the procedures described in "Changing IP addresses and host names" on page 403.

Using multiple token servers

Distributed locking, allowing GPFS to maintain a consistent view of the file system, is implemented using token-based lock management. Associated with every lockable object is a token.

Before a lock on an object can be granted to a thread on a particular node, the lock manager on that node must obtain a token from the token server. The total number of token manager nodes depends on the number of manager nodes defined in the cluster.

When a file system is first mounted, the file system manager is the only token server for the file system. Once the number of external mounts exceeds one, the file system manager appoints all the other manager nodes defined in the cluster to share the token server load. Once the token state has been distributed, it remains distributed until all external mounts have gone away. The only nodes that are eligible to become token manager nodes are those designated as manager nodes.

The number of files for which tokens can be retained on a manager node is restricted by the values of the **maxFilesToCache** and **maxStatCache** configuration parameters. Distributing the tokens across multiple token manager nodes allows more tokens to be managed or retained concurrently, improving performance in situations where many lockable objects are accessed concurrently.

Exporting file system definitions between clusters

You can export a GPFS file system definition from one GPFS cluster to another.

To export file system definitions between clusters, follow these steps:

1. Ensure that all disks in all GPFS file systems to be migrated are in working order by issuing the **mmlsdisk** command. Verify that the disk status is ready and availability is up. If not, correct any problems and reissue the **mmlsdisk** command before continuing.
2. Stop all user activity in the file systems.
3. Follow any local administrative backup procedures to provide for protection of your file system data in the event of a failure.
4. Cleanly unmount all affected GPFS file systems. Do not use force unmount.
5. Export the GPFS file system definitions by issuing the **mmexportfs** command. This command creates the configuration output file *ExportDataFile* with all relevant file system and disk information. Retain this file as it is required when issuing the **mmimportfs** command to import your file systems into the new cluster. Depending on whether you are exporting a single file system or all of the file systems in the cluster, issue:
   ```
   mmexportfs fileSystemName -o ExportDataFile
   ```
   or
   ```
   mmexportfs all -o ExportDataFile
   ```
6. Ensure that the file system disks from the old GPFS cluster are properly connected, and are online and available to be accessed from appropriate nodes of the new GPFS cluster.
7. To complete the movement of your file systems to the new cluster using the configuration file created in Step 5, issue one of these commands, depending on whether you are importing a single file system or all of the file systems in the cluster:
   ```
   mmimportfs fileSystemName -i ExportDataFile
   ```
   or
   ```
   mmimportfs all -i ExportDataFile
   ```

GPFS port usage

The nodes in a GPFS cluster communicate with each other using the TCP/IP protocol. The port number used by the main GPFS daemon (**mmfsd**) is controlled with the **tscTcpPort** configuration parameter. The default port number is 1191.

You can specify a different port number using the **mmchconfig** command:

```
mmchconfig tscTcpPort=PortNumber
```

When the main GPFS daemon (**mmfsd**) is not running on the primary and backup configuration server nodes, a separate service (**mmsdrserv**) is used to provide access to the configuration data to the rest of the nodes in the cluster. The port number used for this purpose is controlled with the **mmsdrservPort** parameter. By default, **mmsdrserv** uses the same port number as the one assigned to the main GPFS daemon. If you change the daemon port number, you must specify the same port number for **mmsdrserv** using the following command:

```
mmchconfig mmsdrservPort=PortNumber
```

Do not change the **mmsdrserv** port number to a number different from that of the daemon port number.

Certain commands (**mmadddisk**, **mmchmgr**, and so on) require an additional socket to be created for the duration of the command. The port numbers assigned to these temporary sockets are controlled with the **tscCmdPortRange** configuration parameter. If an explicit range is not specified, the port number is dynamically assigned by the operating system from the range of ephemeral port numbers. If you want to restrict the range of ports used by GPFS commands, use the **mmchconfig** command:

```
mmchconfig tscCmdPortRange=LowNumber-HighNumber
```

In a remote cluster setup, if GPFS on the remote cluster is configured to use a port number other than the default, you have to specify the port number to be used with the **mmremotecluster** command:

```
mmremotecluster update ClusterName -n tcpPort=PortNumber,Node,Node...
```

Table 59 provides GPFS port usage information:

Table 59. GPFS port usage

Descriptor	Explanation
Service provider	GPFS
Service name	**mmfsd** **mmsdrserv**
Port number	1191 While executing certain commands, GPFS may need to create additional sockets whose dynamic port numbers are assigned by the operating system. Such sockets are used by commands to exchange data with GPFS daemons running on other nodes. The port numbers that are used correspond to the ephemeral ports of the operating system. To control which ports are used by the commands (so that firewall rules can be written to allow incoming traffic only on those ports), you can restrict the port range to a specific range by setting the **tscCmdPortRange** configuration variable.
Protocols	TCP/IP
Source port range	The source port range is chosen by the operating system on the client side.
Is the service name/number pair in the default /etc/services file shipped with AIX and Linux distributions?	See the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
Is the service name/number pair added to /etc/services by a product?	No
Binaries that listen on the ports	`/usr/lpp/mmfs/bin/mmfsd` `/usr/lpp/mmfs/bin/mmsdrserv`

Table 59. GPFS port usage (continued)

Descriptor	**Explanation**
Can the service be configured to use a different port?	Yes. To change the main port used by GPFS, use: `mmchconfig tscTcpPort=`*`PortNumber`* **Note:** If you change the main port (daemon port) number, you must change the **mmsdrserv** port to the same number. To change the **mmsdrserv** port number to match the daemon port number, use: `mmchconfig mmsdrservPort=`*`PortNumber`* To change the range of port numbers used for command execution, use: `mmchconfig tscCmdPortRange=`*`LowNumber-HighNumber`* To specify a port number when connecting to remote clusters, use the **mmremotecluster** command.
When is the service required? What depends on the service?	On the GPFS primary and secondary cluster configuration servers, either **mmsdrserv** or **mmfsd** needs to be running at all times to provide access to GPFS configuration data to the rest of the cluster. On other nodes, **mmfsd** must be running in order to mount a GPFS file system. Depending on the GPFS configuration, a node either has to be a member of the GPFS cluster or possess an authorized SSL key in order to establish a connection.
When the daemon starts and its port is already in use (for example, another resource has bound to it already), how does the daemon behave?	The daemon shuts down and tries to start over again. Most GPFS daemon down error messages are in the **mmfs.log.previous** log for the instance that failed. If the daemon restarted, it generates a new **mmfs.log.latest** log. Begin problem determination for these errors by examining the operating system error log. GPFS records file system or disk failures using the error logging facility provided by the operating system: **syslog** facility on Linux and **errpt** facility on AIX. See the *IBM Spectrum Scale: Problem Determination Guide* for further information.
Is there an administrator interface to query the daemon and have it report its port number?	Yes; run this command: **`mmlsconfig tscTcpPort`**

Is the service/port registered with the Internet Assigned Numbers Authority (IANA)?	Yes
	`gpfs 1191/tcp General Parallel File System`
	`gpfs 1191/udp General Parallel File System`
	`# Dave Craft <gpfs@ibm.com>`
	`November 2004`

Note: Ports configured for **gpfsClusterRemoteShellCommand** (for example, **ssh**) and ICMP (ping) also must be unblocked in the firewall for GPFS to function properly

Securing the IBM Spectrum Scale system using firewall

The IBM Spectrum Scale system is an open system where the customer can interact with the system through other third-party interfaces like MMC, web applications, and so on. The customer also has root access to the system just like any Linux server administrator. Firewalls that are associated with open systems are specific to deployments, operating systems, and it vary from customer to customer. It is the responsibility of the system administrator or Lab Service (LBS) to set the firewall accordingly; similar to what Linux distributions do today. This section provides recommendations to set up firewall to secure the IBM Spectrum Scale protocol nodes.

Firewall recommendations for the IBM Spectrum Scale installation

It is recommended to allow connection only from the IBM Spectrum Scale cluster node IPs (internal IPs and protocol IPs) on port 8889 and block all other external connections on this port during the installation process.

Installer uses the following Chef port during IBM Spectrum Scale installation.

Table 60. Recommended port numbers that can be used for installation

Port Number	Protocol	Service Name	Components involved in communication
8889	TCP	Chef	Intra-cluster and installer server
10080	TCP	Repository	Intra-cluster and installer server

The port that is used during the installation (8889) can be blocked when the installation is over. You can get the list of protocol IPs by using the **mmlscluster --ces** command. Use the **mmlscluster** command to get the list of all internal IPs.

Firewall recommendations for internal communication among nodes

The IBM Spectrum Scale system uses the following ports for internal communication among various IBM Spectrum Scale nodes.

Table 61. Recommended port numbers that can be used for internal communication

Port Number	Protocol	Service Name	Components that are involved in communication
1191	TCP	GPFS	Intra-cluster
22	TCP	SSH	Clients who are accessing the system

The following are the recommendations for securing internal communications among IBM Spectrum Scale nodes:

- Allow connection only to the GPFS cluster node IPs (internal IPs and protocol node IPs) on port 1191. Block all other external connections on this port. Use the **mmlscluster --ces** command to get the list of protocol node IP and use the **mmlscluster** command to get the list of IPs of internal nodes.
- Allow all external communications request that are coming from the admin or management network and IBM Spectrum Scale internal IPs on port 22.
- Certain commands such as **mmadddisk, mmchmgr,** and so on require an extra socket to be created for the duration of the command. The port numbers that are assigned to these temporary sockets are controlled with the **tscCmdPortRange** configuration parameter. If an explicit range is not specified, the

port number is dynamically assigned by the operating system from the range of ephemeral port numbers. It is highly recommended to set the port range. For more information on how to set the port range, see “GPFS port usage” on page 405.

Firewall recommendations for protocol access

It is recommended to use certain port numbers to secure the protocol data transfer.

Recommendations for NFS access

The following table provides the list of static ports that are used for NFS data I/O.

Table 62. Recommended port numbers for NFS access

Port Number	**Protocol**	**Service Name**	**Components that are involved in communication**
2049	TCP and UDP	NFSV4 or NFSV3	NFS clients and IBM Spectrum Scale protocol node
111	TCP and UDP	RPC (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	STATD (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	MNT (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	NLM (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node
User-defined static port	TCP and UDP	RQUOTA (required only by NFSV3)	NFS clients and IBM Spectrum Scale protocol node

Note: The NFSV3 uses the dynamic ports for NLM, MNT, and STATD services. When NFSV4 server is used with the firewall, these services must be configured with static ports.

The following recommendations are applicable:

- Set static ports for MNT, NLM, and STATD services that are required by the NFSV3 server by using **mmnfs configuration change** command. Allow TCP and UDP port 2049 to use the protocol node IPs. For example:

  ```
  mmnfs configuration change MNT_PORT=32767:NLM_PORT=32769:RQUOTA_PORT=32768:
  STATD_PORT=32765
  ```

- Allow all external communications on TCP and UDP port 111 by using the protocol node IPs.
- Allow all external communications on TCP and UDP port that is specified with **mmnfs configuration change** for MNT and NLM ports.

Recommendations for SMB access

Samba uses the following ports for the secure access.

Table 63. Recommended port numbers for SMB access

Port Number	Protocol	Service Name	Components that are involved in communication
445	TCP	Samba	SMB clients and IBM Spectrum Scale protocol node
4379	TCP	CTDB	Inter-protocol node

The following recommendations are applicable for the SMB access:

- Allow the access request that is coming from the data network and admin and management network on port 445 using the protocol node IPs. You can get the list of protocol node IPs by using the **mmlscluster --ces** command.
- Allow connection only to the requests that are coming from the IBM Spectrum Scale cluster node IPs (internal IPs and protocol node IPs) on port 4379. Block all other external connections on this port. Use the **mmlscluster** command to get the list of cluster node IPs.

Object port configuration

Note: IBM Spectrum Scale is configured with the ports listed here. Changing ports requires updating configuration files, Keystone endpoint definitions, and SELinux rules. This must be done only after careful planning.

The following table lists the ports configured for object access.

Table 64. Port numbers for object access

Port Number	Protocol	Service Name	Components that are involved in communication
8080	TCP	Object Storage Proxy	Object clients and IBM Spectrum Scale protocol node
6200	TCP	Object Storage (local account server)	Local host
6201	TCP	Object Storage (local container server)	Local host
6202	TCP	Object Storage (local object server)	Local host
6203	TCP	Object Storage (object server for unified file and object access)	Local host
11211	TCP and UDP	Memcached (local)	Local host

The following ports are configured for securing object access:

- Allow all external communications on TCP port 8080 (Object Storage proxy).
- Allow connection only from the IBM Spectrum Scale cluster node IPs (internal IPs and protocol node IPs) on ports 6200, 6201, 6202, 6203, and 11211. Block all other external connections on this port.

Shell access by non-root users must be restricted on IBM Spectrum Scale protocol nodes where the object services are running to prevent unauthorized access to object data.

Note: The reason for these restrictions is that because there is no authentication of requests made on ports 6200, 6201, 6202, and 6203, it is critical to ensure that these ports are protected from access by unauthorized clients.

Port usage for object authentication

You can configure either an external or internal Keystone server to manage the authentication requests. Keystone uses the following ports:

Table 65. Port numbers for object authentication

Port Number	Protocol	Service Name	Components that are involved in communication
5000	TCP	Keystone Public	Authentication clients and object clients
35357	TCP	Keystone Internal/Admin	Authentication and object clients and Keystone administrator

These ports are applicable only if keystone is hosted internally on the IBM Spectrum Scale system. The following port usage is applicable:
- Allow all external communication requests that are coming from the admin or management network and IBM Spectrum Scale internal IPs on port 35357.
- Allow all external communication requests that are coming from clients to IBM Spectrum Scale for object storage on port 5000. Block all other external connections on this port.

Port usage to connect to the Postgres database for object protocol

The Postgres database server for object protocol is configured to use the following port:

Table 66. Port numbers for Postgres database for object protocol

Port Number	Protocol	Service Name	Components that are involved in communication
5431	TCP and UDP	postgresql-obj	Inter-protocol nodes

It is recommended to allow connection only from Cluster node IPs (Internal ips and Protocol node IPs) on port 5431. Block all other communication requests on this port.

Note: The Postgres instance used by the object protocol uses port 5431. This is different from the default port to avoid conflict with other Postgres instances that might be on the system including the instance for IBM Spectrum Scale GUI.

Consolidated list of recommended ports that are used for installation, internal communication, and protocol access

The following table provides a consolidated list of recommended ports and firewall rules.

Table 67. Consolidated list of recommended ports for different functions

Function	Dependent network service names	External ports that are used for file and object access	Internal ports that are used for inter-cluster communication	UDP / TCP	Nodes for which the rules are applicable
Installer	Chef	N/A	8889 (chef) 10080 (repo)	TCP	GPFS server, NSD server, protocol nodes
GPFS (internal communication)	GPFS	N/A	1191 (GPFS) 60000-61000 for tscCmdPortRange 22 for SSH	TCP and UDP TCP only for 22	GPFS server, NSD server, protocol nodes
SMB	gpfs-smb.service gpfs-ctdb.service rpc.statd	445	4379 (CTDB)	TCP	Protocol nodes only
NFS	ganesha.nfsd rpcbind rpc.statd	2049 (NFS_PORT - required only by NFSV3) 111 (RPC - required only by NFSV3) 32765 (STATD_PORT) 32767 (MNT_PORT - required only by NFSV3) 32768 (RQUOTA_PORT - required only by NFSV3) 32769 (NLM_PORT - required only by NFSV3) **Note:** Make the dynamic ports as static with command `mmnfs configuration change` .	N/A	TCP and UDP	Protocol nodes only
Object	swift-proxy-server keystone-all postgresql-obj	8080 (proxy server) 35357 (keystone) 5000 (keystone public)	5431 (Object Postgres instance) 6200-6203 (Object Storage) 11211 (Memcached)	TCP TCP and UDP (for 11211 only)	Protocol nodes only

Firewall recommendations for IBM Spectrum Scale GUI

Dedicating certain ports for firewalls helps to secure IBM Spectrum Scale management and install GUIs. Different ports are used for securing installation GUI and management GUI.

The following table lists the ports that need to be used to secure GUI.

Table 68. Firewall recommendations for GUI

Port Number	GUI Type	Protocol
9080	Installation	HTTP
9443	Installation	HTTPS
80	Management	HTTP
443	Management	HTTPS

The management GUI uses ZIMon to collect performance data. ZIMon collectors are normally deployed with the management GUI and sometimes on other systems in a federated configuration.

Each ZIMon collector uses three ports, which can be configured in ZIMonCollector.cfg. The default ports are 4739, 9085, and 9084.

Firewall recommendations for Performance Monitoring tool

The IBM Spectrum Scale system uses the following ports for the Performance Monitoring tool to work.

Table 69. Recommended port numbers that can be used for Performance Monitoring tool

Port Number	Protocol	Service Name	Components that are involved in communication
4739	TCP and UDP	Performance Monitoring tool	Intra-cluster
8123	TCP	Object Metric collection	Intra-cluster
8124	TCP	Object Metric collection	Intra-cluster
8125	UDP	Object Metric collection	Intra-cluster
8126	TCP	Object Metric collection	Intra-cluster
8127	TCP	Object Metric collection	Intra-cluster
9084	TCP	Performance Monitoring Tool	Any node that wants to query the database
9085	TCP	Performance Monitoring Tool	Intra-cluster

Important:

- The 4739 port needs to be open when a collector is installed.
- The 9085 port needs to be open when there are two or more collectors.
- If the 9084 port is closed, accessing the collector to debug or to connect external tools or, even another instance of the GUI, remotely is not possible, except from the node where the GUI and the collector are installed.

Supported web browser versions and web browser settings for GUI

To access the management GUI, you must ensure that your web browser is supported and has the appropriate settings enabled.

The management GUI supports the following web browsers:
- Mozilla Firefox 41
- Mozilla Firefox Extended Support Release (ESR) 38
- Microsoft Internet Explorer (IE) 10 and 11
- Google Chrome 45

IBM® supports higher versions of the browsers if the vendors do not remove or disable function that the product relies upon. For browser levels higher than the versions that are certified with the product, customer support accepts usage-related and defect-related service requests. If the support center cannot re-create the issue, support might request the client to re-create the problem on a certified browser version. Defects are not accepted for cosmetic differences between browsers or browser versions that do not affect the functional behavior of the product. If a problem is identified in the product, defects are accepted. If a problem is identified with the browser, IBM might investigate potential solutions or work-arounds that the client can implement until a permanent solution becomes available.

To configure your web browser, follow these steps:
1. Enable JavaScript for your web browser.

   For Mozilla Firefox, JavaScript is enabled by default and requires no additional configuration.

   For Microsoft Internet Explorer (IE) running on Microsoft Windows 7:
   a. In Internet Explorer, click **Tools** > **Internet Options**.
   b. Click **Security Settings**.
   c. Click **Internet** to choose the Internet zone.
   d. Click **Custom Level**.
   e. Scroll down to the **Scripting** section, and then in **Active Scripting**, click **Enable**.
   f. Click **OK** to close **Security Settings**.
   g. Click **Yes** to confirm the change for the zone.
   h. Click **OK** to close **Internet Options**.
   i. Refresh your browser.

   For Microsoft Internet Explorer (IE) running on Microsoft Windows Server 2008:
   a. In Internet Explorer, click **Tools** > **Internet Options**.
   b. Click **Security**.
   c. Click **Trusted sites**.
   d. On the **Trusted sites** dialog, verify that the web address for the management GUI is correct and click **Add**.
   e. Verify that the correct web address was added to the **Trusted sites** dialog.
   f. Click **Close** on the **Trusted sites** dialog.
   g. Click **OK**.
   h. Refresh your browser.

   For Google Chrome:
   a. On the menu bar in the Google Chrome browser window, click **Settings**.
   b. Click **Show advanced settings**.
   c. In the **Privacy** section, click **Content settings**.
   d. In the **JavaScript** section, select **Allow all sites to run JavaScript**.

e. Click **OK**.

f. Refresh your browser.

2. Enable cookies in your web browser.

   For Mozilla Firefox:

   a. On the menu bar in the Firefox browser window, click **Tools** > **Options**.

   b. On the Options window, select **Privacy**.

   c. Set "Firefox will" to **Use custom settings for history**.

   d. Select **Accept cookies from sites** to enable cookies.

   e. Click **OK**.

   f. Refresh the browser.

   For Microsoft Internet Explorer:

   a. In Internet Explorer, click **Tools** > **Internet Options**.

   b. Click **Privacy**. Under **Settings**, move the slider to the bottom to allow all cookies.

   c. Click **OK**.

   d. Refresh your browser.

   For Google Chrome:

   a. On the menu bar in the Google Chrome browser window, click **Settings**.

   b. Click **Show advanced settings**.

   c. In the **Privacy** section, click **Content settings**.

   d. In the **Cookies** section, select **Allow local data to be set**.

   e. Click **OK**.

   f. Refresh your browser.

3. Enable file download on IE 10 and 11 running on Windows 2012.

   a. In Internet Explorer, click **Tools** > **Internet Options**.

   b. On the Internet Options window, select the **Security** tab.

   c. On the **Security** tab, click the **Internet zone**.

   d. Click **Custom level** to customize the security level for this zone.

   e. Scroll down to **Downloads** and select **Enable** under File download.

   f. Click **OK**.

   g. Click **Yes** to confirm.

   h. Click **OK** to close the Internet Options.

4. Enable scripts to disable or replace context menus. (Mozilla Firefox only).

   For Mozilla Firefox:

   a. On the menu bar in the Firefox browser window, click **Tools** > **Options**.

   b. On the Options window, select **Content**.

   c. Click **Advanced** by the **Enable JavaScript** setting.

   d. Select **Disable or replace context menus**.

   e. Click **OK** to close the Advanced window.

   f. Click **OK** to close the Options window.

   g. Refresh your browser.

Chapter 25. GUI limitations

The following are the limitations of the IBM Spectrum Scale GUI:

1. The GUI supports only RHEL7.x or SLES12 as the operating system on Power (Big or Little Endian) or Intel x86. However, other nodes in the IBM Spectrum Scale cluster could be on other platforms and operating systems.
2. Up to 128 nodes are supported.
3. The GUI supports a subset of the CLI functionality. Additional capabilities will be added in the future releases of the product.
4. The Object management panels do not support configurations with Keystone V2 API, HTTPS communication to Keystone, and AD/LDAP-backed Keystone configurations.
5. One GUI instance supports a single cluster.
6. The GUI does not support file system creation. You need to use the CLI to create file system.
7. Snapshots are deleted only once in a day to avoid performance issues. Due to this limitation, it is possible that there might be certain discrepancy between the number of snapshots present on the system and the number that should remain according to the snapshot retention rules.
8. In an IBM Spectrum Scale and Elastic Storage Server (ESS) mixed support environment, ESS GUI must manage the whole cluster to not to lose the ESS-specific pages in the GUI. You must not setup ESS GUI on a protocol node.
9. The GUI does not support GNR-based solutions. The GNR-based solutions have their own separate GUI to deploy and mange that system.

| For limitations of the installation GUI, see *Installing IBM Spectrum Scale by using the graphical user interface*
| *(GUI)* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Chapter 26. Understanding the call home functionality

IBM Spectrum Scale introduces two call home features. The first feature is to collect predefined data from each GPFS node on a regular basis and upload this data to IBM support center. This data can be used by the support or development personnel for problem analysis.

The second feature is to upload a specific file to IBM support center.

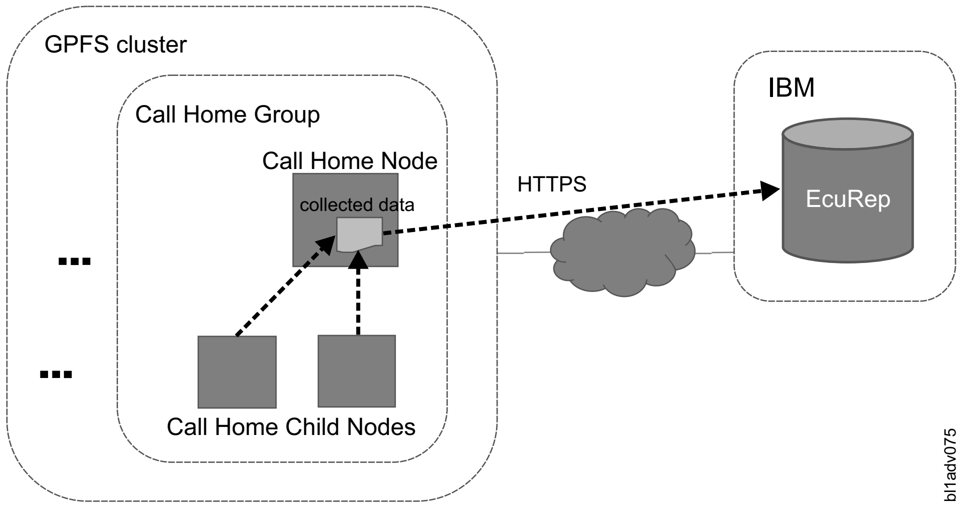


Figure 17. Call home architecture

IBM Spectrum Scale supports two predefined data collection tasks: daily and weekly.

Call home terminologies:

The terminologies associated with the call home feature are listed here:

- Call home group: A group of nodes configured by using the **mmcallhome group** command which consists of one call home node and multiple call home child nodes. Multiple call home groups can be configured within a GPFS cluster.
- Call home node: This node initiates the data collection within the call home group and uploads the data package to IBM support center.
- Call home child node: A node that the call home node collects data from. The call home node is also determined as its child node.
- Gather-Send task: A gather-send task is a process that runs on the call home node to collect data from the child nodes and upload the data to the configured server. The gather-send configuration file will include information about what needs to be collected from the child nodes.
- Send file task: A process which runs on the call home node to upload a file specified by a user.
- Data package directory: A directory where the data packages will be placed and stored after being uploaded for backup.

Requirements/Prerequisites:

The following criteria must be met to be able to use the call home functionality on your GPFS cluster:

- Only Linux nodes are supported.

- Only x86 or Power PC big endian platform is supported.
- Only supported on RHEL 7.
- At least one call home group needs to be defined.
- GPFS CCR needs to be enabled
- The call home node needs to be able to access the following IP addresses and ports:
  - Host name: esupport.ibm.com
  - IP address: 129.42.56.189, 129.42.60.189, 129.42.52.189
  - Port number: 443

  The recommendation is to open 129.42.0.0/18
- The call home node needs to have 1 GB free space under the data package directory (2 - 3 GB preferred)
- There must not be multiple nodes with the same short host name within the GPFS cluster.
- Only GPFS cluster with /usr/bin/scp (or any other path to scp) configured for the file transfer method is supported.
- Install the call home RPMs in the following order:
  1. gpfs.callhome-jre-8.0-1.0
  2. gpfs.callhome-ecc-client-4.2-1.0
  3. gpfs.callhome-4.2-1
- Ensure that the following dependencies are resolved when you install call home:
  - Dependencies for gpfs.callhome-jre-8.0-1.0:
    - libXext
    - libXi
    - libXtst
  - Dependencies for gpfs.callhome-4.2 and above:
    - perl-List-MoreUtils
    - perl-Module-Runtime
    - perl-Try-Tiny
    - perl-Module-Implementation
    - perl-Params-Validate
    - perl-DateTime-Locale
    - perl-Sub-Install
    - perl-Params-Util
    - perl-Data-OptList
    - perl-Package-DeprecationManager
    - perl-Package-Stash-XS
    - perl-Package-Stash
    - perl-Class-Load
    - perl-Class-Singleton
    - perl-TimeDate
    - perl-DateTime
    - perl-DateTime-TimeZone
    - perl-Data-Dumper package

Data package file

Every time call home collects data or the call home command is invoked to upload a specific file, call home first creates a data package file. The data package file is a tar file. The successfully uploaded files are saved on the call home node under `<gpfs data dump dir>/callhome/rsENUploaded` (<gpfs data dump dir> is a GPFS global define) for backup.

Initial Configuration

By default, the call home functionality is disabled.

To enable the functionality, the administrator needs to take the following actions:
- Configure the call home groups
- Set the customer name, customer ID, and the country code
- Enable the call home capability
- Add the tasks required for periodic data collection
- If a specific file needs to be uploaded, execute the **mmcallhome run** command

The **mmcallhome** CLI provides interface to configure, enable, run, schedule, and monitor call home related tasks in the GPFS cluster. See the **mmcallhome** man page for more information.

Below are some examples for configuring, enabling, running, scheduling and monitoring call home related tasks.

1. To configure a call home group, issue this command:
   ```
   mmcallhome group add group1 themisto0 -N themisto0,themisto1,themisto2
   ```
   The system displays output similar to this:
   ```
   Call home group group1 has been created
   ```
2. To change customer information such as customer name, customer ID, and the country code, issue this command:
   ```
   mmcallhome info change --customer-name "SpectrumScaleTest" --customerid
   "1234" --country-code "JP"
   ```
   The system displays output similar to this:
   ```
   Success
   ```
3. To view the customer information, issue this command:
   ```
   mmcallhome info list
   ```
   The system displays output similar to this:
   ```
   Parameter                             Value
   customer-name                         SpectrumScaleTest
   customer-id                           1234
   voice-phone
   offshift-phone
   modem-phone
   modem-prefix
   ign1phone
   ign2phone
   customer-location
   email
   special-instruction
   callhome-method                       ethernet
   remote-service-callback-number
   country-code                          JP
   ```
4. To enable the call home service, issue this command:
   ```
   mmcallhome capability enable
   ```

The system displays output similar to this:

```
Call home node: themisto0
Call home child nodes to collect data: themisto0 themisto1 themisto2 (total 3 nodes)
Excluded nodes:
SSH Access Check: OK
Data package directory: /tmp/mmfs/callhome
Success
```

5. To register a daily task with cron, issue this command:

   ```
   mmcallhome schedule add --task daily
   ```

   The system displays output similar to this:

   ```
   /etc/cron.d/gpfscallhome_gatherSend_daily.conf registered
   41 command entries are defined for this task
   ```

6. To register a weekly task with cron, issue this command:

   ```
   mmcallhome schedule add --task weekly
   ```

   The system displays output similar to this:

   ```
   /etc/cron.d/gpfscallhome_gatherSend_weekly.conf registered
   14 command entries are defined for this task
   ```

7. To list the registered tasks for gather-send, issue this command:

   ```
   mmcallhome schedule list
   ```

   The system displays output similar to this:

   ```
   Registered Tasks for gatherSend:
   ConfFile               CronParameters
   daily.conf             3 2 * * *
   weekly.conf            54 3 * * sun
   ```

8. To monitor the call home tasks, issue this command:

   ```
   mmcallhome status list
   ```

   The system displays output similar to this:

   ```
   Task   Start time          Status       Package file name
   daily  20150930132656.582  success      ...aultDaily.g_daily.20150930132656582.cl0.DC
   daily  20150930133134.802  success      ...aultDaily.g_daily.20150930133134802.cl0.DC
   daily  20150930133537.509  success      ...aultDaily.g_daily.20150930133537509.cl0.DC
   daily  20150930133923.063  success      ...aultDaily.g_daily.20150930133923063.cl0.DC
   RunSendFile 20150930133422.843 success
   ...group2.MyTestData.s_file.20150930133422843.cl0.DC
   ```

9. To view the status of the currently running and the already completed call home tasks, issue this command:

   ```
   mmcallhome status list --verbose
   ```

   The system displays output similar to this:

   ```
   Task Start time Updated time Status RC or Step
   Package file name
   [ additional info: value ]
   --------------------------------------------------------------------------------
   RunSendFile 20150930193046.693 20150930193059 success RC=0
   11786648094375.4_2_0_0.1234.SpectrumScaleTest.group1.MyTestData.s_fil
   e.20150930193046693.cl0.DC
   Original file name: testFile
   --------------------------------------------------------------------------------
   ```

Accessibility features for IBM Spectrum Scale

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in IBM Spectrum Scale:
- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

IBM Knowledge Center, and its related publications, are accessibility-enabled. The accessibility features are described in IBM Knowledge Center (www.ibm.com/support/knowledgecenter).

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

IBM and accessibility

See the IBM Human Ability and Accessibility Center (www.ibm.com/able) for more information about the commitment that IBM has to accessibility.

Notices

This information was developed for products and services that are offered in the USA.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Trademarks

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering’s use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM’s Privacy Policy at http://www.ibm.com/privacy and IBM’s Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled “Cookies, Web Beacons and Other Technologies” and the “IBM Software Products and Software-as-a-Service Privacy Statement” at http://www.ibm.com/software/info/product-privacy.

Glossary

This glossary provides terms and definitions for IBM Spectrum Scale.

The following cross-references are used in this glossary:

- *See* refers you from a nonpreferred term to the preferred term or from an abbreviation to the spelled-out form.
- *See also* refers you to a related or contrasting term.

For other terms and definitions, see the IBM Terminology website (www.ibm.com/software/globalization/terminology) (opens in new window).

B

block utilization
: The measurement of the percentage of used subblocks per allocated blocks.

C

cluster
: A loosely-coupled collection of independent systems (nodes) organized into a network for the purpose of sharing resources and communicating with each other. See also *GPFS cluster*.

cluster configuration data
: The configuration data that is stored on the cluster configuration servers.

cluster manager
: The node that monitors node status using disk leases, detects failures, drives recovery, and selects file system managers. The cluster manager must be a quorum node. The selection of the cluster manager node favors the quorum-manager node with the lowest node number among the nodes that are operating at that particular time.

  Note: The cluster manager role is not moved to another node when a node with a lower node number becomes active.

control data structures
: Data structures needed to manage file data and metadata cached in memory. Control data structures include hash tables and link pointers for finding cached data; lock states and tokens to implement distributed locking; and various flags and sequence numbers to keep track of updates to the cached data.

D

Data Management Application Program Interface (DMAPI)
: The interface defined by the Open Group's XDSM standard as described in the publication *System Management: Data Storage Management (XDSM) API Common Application Environment (CAE) Specification C429*, The Open Group ISBN 1-85912-190-X.

deadman switch timer
: A kernel timer that works on a node that has lost its disk lease and has outstanding I/O requests. This timer ensures that the node cannot complete the outstanding I/O requests (which would risk causing file system corruption), by causing a panic in the kernel.

dependent fileset
: A fileset that shares the inode space of an existing independent fileset.

disk descriptor
: A definition of the type of data that the disk contains and the failure group to which this disk belongs. See also *failure group*.

disk leasing
: A method for controlling access to storage devices from multiple host systems. Any host that wants to access a storage device configured to use disk leasing registers for a lease; in the event of a perceived failure, a host system can deny access, preventing I/O operations with the storage device until the preempted system has reregistered.

disposition
: The session to which a data management event is delivered. An individual disposition is set for each type of event from each file system.

domain
A logical grouping of resources in a network for the purpose of common management and administration.

E

ECKD™
See *extended count key data (ECKD).*

ECKD device
See *extended count key data device (ECKD device).*

encryption key
A mathematical value that allows components to verify that they are in communication with the expected server. Encryption keys are based on a public or private key pair that is created during the installation process. See also *file encryption key*, *master encryption key*.

extended count key data (ECKD)
An extension of the count-key-data (CKD) architecture. It includes additional commands that can be used to improve performance.

extended count key data device (ECKD device)
A disk storage device that has a data transfer rate faster than some processors can utilize and that is connected to the processor through use of a speed matching buffer. A specialized channel program is needed to communicate with such a device. See also *fixed-block architecture disk device*.

F

failback
Cluster recovery from failover following repair. See also *failover*.

failover
(1) The assumption of file system duties by another node when a node fails. (2) The process of transferring all control of the ESS to a single cluster in the ESS when the other clusters in the ESS fails. See also *cluster*. (3) The routing of all transactions to a second controller when the first controller fails. See also *cluster*.

failure group
A collection of disks that share common access paths or adapter connection, and could all become unavailable through a single hardware failure.

FEK See *file encryption key*.

fileset A hierarchical grouping of files managed as a unit for balancing workload across a cluster. See also *dependent fileset*, *independent fileset*.

fileset snapshot
A snapshot of an independent fileset plus all dependent filesets.

file clone
A writable snapshot of an individual file.

file encryption key (FEK)
A key used to encrypt sectors of an individual file. See also *encryption key*.

file-management policy
A set of rules defined in a policy file that GPFS uses to manage file migration and file deletion. See also *policy*.

file-placement policy
A set of rules defined in a policy file that GPFS uses to manage the initial placement of a newly created file. See also *policy*.

file system descriptor
A data structure containing key information about a file system. This information includes the disks assigned to the file system (*stripe group*), the current state of the file system, and pointers to key files such as quota files and log files.

file system descriptor quorum
The number of disks needed in order to write the file system descriptor correctly.

file system manager
The provider of services for all the nodes using a single file system. A file system manager processes changes to the state or description of the file system, controls the regions of disks that are allocated to each node, and controls token management and quota management.

fixed-block architecture disk device (FBA disk device)
A disk device that stores data in blocks of fixed size. These blocks are addressed by block number relative to the beginning of the file. See also *extended count key data device*.

fragment
The space allocated for an amount of data

too small to require a full block. A fragment consists of one or more subblocks.

G

global snapshot
A snapshot of an entire GPFS file system.

GPFS cluster
A cluster of nodes defined as being available for use by GPFS file systems.

GPFS portability layer
The interface module that each installation must build for its specific hardware platform and Linux distribution.

GPFS recovery log
A file that contains a record of metadata activity, and exists for each node of a cluster. In the event of a node failure, the recovery log for the failed node is replayed, restoring the file system to a consistent state and allowing other nodes to continue working.

I

ill-placed file
A file assigned to one storage pool, but having some or all of its data in a different storage pool.

ill-replicated file
A file with contents that are not correctly replicated according to the desired setting for that file. This situation occurs in the interval between a change in the file's replication settings or suspending one of its disks, and the restripe of the file.

independent fileset
A fileset that has its own inode space.

indirect block
A block containing pointers to other blocks.

inode The internal structure that describes the individual files in the file system. There is one inode for each file.

inode space
A collection of inode number ranges reserved for an independent fileset, which enables more efficient per-fileset functions.

ISKLM
IBM Security Key Lifecycle Manager. For GPFS encryption, the ISKLM is used as an RKM server to store MEKs.

J

journaled file system (JFS)
A technology designed for high-throughput server environments, which are important for running intranet and other high-performance e-business file servers.

junction
A special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

K

kernel The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.

M

master encryption key (MEK)
A key used to encrypt other keys. See also *encryption key*.

MEK See *master encryption key*.

metadata
Data structures that contain information that is needed to access file data. Metadata includes inodes, indirect blocks, and directories. Metadata is not accessible to user applications.

metanode
The one node per open file that is responsible for maintaining file metadata integrity. In most cases, the node that has had the file open for the longest period of continuous time is the metanode.

mirroring
The process of writing the same data to multiple disks at the same time. The mirroring of data protects it against data loss within the database or within the recovery log.

multi-tailed
A disk connected to multiple nodes.

N

namespace
Space reserved by a file system to contain the names of its objects.

Network File System (NFS)
A protocol, developed by Sun Microsystems, Incorporated, that allows any host in a network to gain access to another host or netgroup and their file directories.

Network Shared Disk (NSD)
A component for cluster-wide disk naming and access.

NSD volume ID
A unique 16 digit hex number that is used to identify and access all NSDs.

node An individual operating-system image within a cluster. Depending on the way in which the computer system is partitioned, it may contain one or more nodes.

node descriptor
A definition that indicates how GPFS uses a node. Possible functions include: manager node, client node, quorum node, and nonquorum node.

node number
A number that is generated and maintained by GPFS as the cluster is created, and as nodes are added to or deleted from the cluster.

node quorum
The minimum number of nodes that must be running in order for the daemon to start.

node quorum with tiebreaker disks
A form of quorum that allows GPFS to run with as little as one quorum node available, as long as there is access to a majority of the quorum disks.

non-quorum node
A node in a cluster that is not counted for the purposes of quorum determination.

P

policy A list of file-placement, service-class, and encryption rules that define characteristics and placement of files. Several policies can be defined within the configuration, but only one policy set is active at one time.

policy rule
A programming statement within a policy that defines a specific action to be performed.

pool A group of resources with similar characteristics and attributes.

portability
The ability of a programming language to compile successfully on different operating systems without requiring changes to the source code.

primary GPFS cluster configuration server
In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data.

private IP address
A IP address used to communicate on a private network.

public IP address
A IP address used to communicate on a public network.

Q

quorum node
A node in the cluster that is counted to determine whether a quorum exists.

quota The amount of disk space and number of inodes assigned as upper limits for a specified user, group of users, or fileset.

quota management
The allocation of disk blocks to the other nodes writing to the file system, and comparison of the allocated space to quota limits at regular intervals.

R

Redundant Array of Independent Disks (RAID)
A collection of two or more disk physical drives that present to the host an image of one or more logical disk drives. In the event of a single physical device failure, the data can be read or regenerated from the other disk drives in the array due to data redundancy.

recovery
The process of restoring access to file system data when a failure has occurred. Recovery can involve reconstructing data or providing alternative routing through a different server.

remote key management server (RKM server)
A server that is used to store master encryption keys.

replication
The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target), and synchronizing the data in both locations.

RKM server
See *remote key management server*.

rule A list of conditions and actions that are triggered when certain conditions are met. Conditions include attributes about an object (file name, type or extension, dates, owner, and groups), the requesting client, and the container name associated with the object.

S

SAN-attached
Disks that are physically attached to all nodes in the cluster using Serial Storage Architecture (SSA) connections or using Fibre Channel switches.

Scale Out Backup and Restore (SOBAR)
A specialized mechanism for data protection against disaster only for GPFS file systems that are managed by Tivoli Storage Manager (TSM) Hierarchical Storage Management (HSM).

secondary GPFS cluster configuration server
In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data in the event that the primary GPFS cluster configuration server fails or becomes unavailable.

Secure Hash Algorithm digest (SHA digest)
A character string used to identify a GPFS security key.

session failure
The loss of all resources of a data management session due to the failure of the daemon on the session node.

session node
The node on which a data management session was created.

Small Computer System Interface (SCSI)
An ANSI-standard electronic interface that allows personal computers to communicate with peripheral hardware, such as disk drives, tape drives, CD-ROM drives, printers, and scanners faster and more flexibly than previous interfaces.

snapshot
An exact copy of changed data in the active files and directories of a file system or fileset at a single point in time. See also *fileset snapshot*, *global snapshot*.

source node
The node on which a data management event is generated.

stand-alone client
The node in a one-node cluster.

storage area network (SAN)
A dedicated storage network tailored to a specific environment, combining servers, storage products, networking products, software, and services.

storage pool
A grouping of storage space consisting of volumes, logical unit numbers (LUNs), or addresses that share a common set of administrative characteristics.

stripe group
The set of disks comprising the storage assigned to a file system.

striping
A storage process in which information is split into blocks (a fixed amount of data) and the blocks are written to (or read from) a series of disks in parallel.

subblock
The smallest unit of data accessible in an I/O operation, equal to one thirty-second of a data block.

system storage pool
A storage pool containing file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks and extended attributes The **system storage pool** can also contain user data.

T

token management
A system for controlling file access in which each application performing a read or write operation is granted some form of access to a specific block of file data.

Token management provides data consistency and controls conflicts. Token management has two components: the token management server, and the token management function.

token management function
A component of token management that requests tokens from the token management server. The token management function is located on each cluster node.

token management server
A component of token management that controls tokens relating to the operation of the file system. The token management server is located at the file system manager node.

twin-tailed
A disk connected to two nodes.

U

user storage pool
A storage pool containing the blocks of data that make up user files.

V

VFS See *virtual file system*.

virtual file system (VFS)
A remote file system that has been mounted so that it is accessible to the local user.

virtual node (vnode)
The structure that contains information about a file system object in a virtual file system (VFS).

Index

Product Number: 5725-Q01
5641-GPF
5725-S28

Printed in USA

SC23-7032-06