IBM Spectrum Scale
Version 4 Release 2.0

Concepts, Planning, and Installation Guide

GA76-0441-06

IBM Spectrum Scale
Version 4 Release 2.0

Concepts, Planning, and Installation Guide

GA76-0441-06

Note
Before using this information and the product it supports, read the information in "Notices" on page 207.

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 xii. 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.

© Copyright IBM Corporation 2014, 2016.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2014, 2016

Figures

© Copyright IBM Corp. 2014, 2016

Tables

© Copyright IBM Corp. 2014, 2016

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

© Copyright IBM Corp. 2014, 2016

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 xii 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:

© Copyright IBM Corp. 2014, 2016

- *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. Introducing IBM Spectrum Scale

IBM Spectrum Scale product structure

IBM Spectrum Scale has three different editions based on functional levels. Each edition can be licensed by IBM Spectrum Scale Client license, IBM Spectrum Scale Server license, and IBM Spectrum Scale FPO license.

IBM Spectrum Scale Express Edition
Available on AIX, Linux (including Linux on z Systems), and Windows. Provides the base GPFS functions.

IBM Spectrum Scale Standard Edition
Available on AIX, Linux (including Linux on z Systems), and Windows. Provides extended features in addition to the base GPFS functions provided in the IBM Spectrum Scale Express Edition. On AIX and Linux, the extended features include Information Lifecycle Management (ILM), Active File Management (AFM), and Clustered NFS (CNFS). CNFS is not available on Linux on z Systems. On Windows, the extended features include limited Information Lifecycle Management (ILM).

On Red Hat Enterprise Linux 7, the extended features include the ability to enable and use the additional protocol functionality integration provided with this release. Two packages are provided for this edition: one that includes protocols and needs additional (opensource/GPL) license acceptance, and another that does not include protocols and requires the traditional GPFS license acceptance.

IBM Spectrum Scale Advanced Edition
Available on AIX and Linux. (File compression is not supported on Linux on z Systems). Provides all the features of the Standard Edition and also provides AFM asynchronous data replication and high-level data protection using the GPFS cryptographic subsystem. For additional information, see the *Encryption* topic in *IBM Spectrum Scale: Advanced Administration Guide*.

Note: All nodes in a cluster must have the same edition installed.

Consult the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) for the latest features included in each edition.

Overview of GPFS

GPFS is a cluster file system that provides concurrent access to a single file system or set of file systems from multiple nodes. The nodes can be SAN attached, network attached, a mixture of SAN attached and network attached, or in a shared nothing cluster configuration. This enables high performance access to this common set of data to support a scale-out solution or provide a high availability platform.

GPFS has many features beyond common data access including data replication, policy based storage management, and multi-site operations. You can create a GPFS cluster of AIX nodes, Linux nodes, Windows server nodes, or a mix of all three. GPFS can run on virtualized instances providing common data access in environments, leverage logical partitioning, or other hypervisors. Multiple GPFS clusters can share data within a location or across wide area network (WAN) connections.

The strengths of GPFS

GPFS provides a global namespace, shared file system access among GPFS clusters, simultaneous file access from multiple nodes, high recoverability and data availability through replication, the ability to make changes while a file system is mounted, and simplified administration even in large environments.

© Copyright IBM Corporation © IBM 2014, 2016

For more information, see the following:

Shared file system access among GPFS clusters

GPFS allows you to share data between separate clusters within a location or across a WAN. Between clusters, users can share access to some or all file systems in either cluster.

Access between clusters can be used for administrative purposes; for example a cluster may not own any storage, but may mount file systems only from another cluster. This configuration allows for separate management of GPFS clusters. This configuration is referred to as a multi-cluster environment using the GPFS remote mount feature. When multiple clusters are configured to access the same GPFS file system, IBM Global Security Kit (GSKit) can be used to authenticate and check authorization for all network connections.

GSKit can be used for authentication and to encrypt data passed between the clusters. If you use a GSKit cipher, the data is encrypted for transmissions. However, if you set the **cipherlist** keyword of the **mmauth** command to **AUTHONLY**, GSKit is only used for authentication and not for data transmissions.

Note: If you have a GPFS 4.1 cluster and a GPFS 3.5 cluster, you can use GSKit on the 4.1 cluster and Open Secure Socket Layer (OpenSSL) on the 3.5 cluster.

GPFS multi-cluster environment provides for:

- The ability of the cluster hosting the file system to specify different security levels, for each cluster authorized to mount a particular file system.
- The local cluster may remain active while changing security keys. Periodic changing of keys is necessary for a variety of reasons, including:
  - In order to make connection rate performance acceptable in large clusters, the size of the security keys used for authentication cannot be very large. As a result it may be necessary to change security keys in order to prevent a given key from being compromised while it is still in use.
  - As a matter of policy, some institutions may require security keys are changed periodically.

The pair of public and private security keys provided by GPFS are similar to the host-based authentication mechanism provided by OpenSSH. Each GPFS cluster has a pair of these keys that identify the cluster. In addition, each cluster also has an `authorized_keys` list. Each line in the `authorized_keys` list contains the public key of one remote cluster and a list of file systems that cluster is authorized to mount. For details about multi-cluster (remote mount) file system access, see *Accessing GPFS file systems from other GPFS clusters* in *IBM Spectrum Scale: Advanced Administration Guide*.

See also the *Active file management* chapter in the *IBM Spectrum Scale: Advanced Administration Guide*.

Improved system performance

Using GPFS file systems can improve system performance in the following ways.

- Allowing multiple processes or applications on all nodes in the cluster simultaneous access to the same file using standard file system calls.
- Increasing aggregate bandwidth of your file system by spreading reads and writes across multiple disks.

- Balancing the load evenly across all disks to maximize their combined throughput, eliminating storage hotspots.
- Supporting very large file and file system sizes.
- Allowing concurrent reads and writes from multiple nodes.
- Allowing for distributed token (lock) management. Distributing token management reduces system delays associated with a lockable object waiting to obtaining a token. Refer to "Increased data availability" for additional information on token management.
- Allowing for the specification of multiple networks for GPFS daemon communication and for GPFS administration command usage within your cluster.

Achieving high throughput to a single, large file requires striping data across multiple disks and multiple disk controllers. Rather than relying on striping in a separate volume manager layer, GPFS implements striping in the file system. Managing its own striping affords GPFS the control it needs to achieve fault tolerance and to balance load across adapters, storage controllers, and disks. Large files in GPFS are divided into equal sized blocks, and consecutive blocks are placed on different disks in a round-robin fashion.

To exploit disk parallelism when reading a large file from a single-threaded application, whenever it can recognize a pattern, GPFS intelligently prefetches data into its buffer pool, issuing I/O requests in parallel to as many disks as necessary to achieve the peak bandwidth of the underlying storage hardware infrastructure. GPFS recognizes multiple I/O patterns including sequential, reverse sequential, and various forms of strided access patterns.

GPFS I/O performance may be monitored through the **mmpmon** command. For more information, see *mmpmon command* in *IBM Spectrum Scale: Administration and Programming Reference*.

File consistency

GPFS uses a sophisticated token management system to provide data consistency while allowing multiple independent paths to the same file by the same name from anywhere in the cluster.

For more information, see "GPFS architecture" on page 9.

Increased data availability

GPFS provides multiple features that improve the reliability of your file system. This includes automatic features like file system logging and configurable features like intelligently mounting file systems on startup to providing tools for flexible synchronous replication.

GPFS allows you to organize your storage hardware into *failure groups*. A failure group is defined as a set of disks that share a common point of failure that could cause them all to become simultaneously unavailable. Failure groups are defined by the system administrator, so care needs to be taken when defining disks to ensure proper failure group isolation. When used in conjunction with the *replication* feature of GPFS, the creation of multiple failure groups provides for increased file availability should a group of disks fail. Replication in GPFS ensures that there is a copy of each block of replicated data and metadata on disks in different failure groups. In this case, should a set of disks become unavailable, GPFS fails over to the replicated copies in another failure group.

During configuration, you assign a replication factor to indicate the total number of copies of data and metadata you wish to store. Currently the maximum replication factor is 3, indicating that three copies of data and metadata should be kept for replicated files. Replication allows you to set different levels of protection for each file or one level for an entire file system. Since replication uses additional disk space and requires extra write time, you should consider the impact of replication on your application, especially if the replication is occurring over a WAN. To reduce the overhead involved with the replication of data, you may also choose to replicate only metadata as a means of providing additional file system protection. For further information on GPFS replication, see "File system replication parameters" on page 63.

GPFS is a logging file system and creates separate logs for each node. These logs record the allocation and modification of metadata aiding in fast recovery and the restoration of data consistency in the event of node failure. Even if you do not specify replication when creating a file system, GPFS automatically replicates recovery logs in separate failure groups, if multiple failure groups are available. This replication feature can be used in conjunction with other GPFS capabilities to maintain one replica in a geographically separate location which provides the capability to survive disasters at the other location. For further information on failure groups, see "Network Shared Disk (NSD) creation considerations" on page 52. For further information on disaster recovery with GPFS see the *IBM Spectrum Scale: Advanced Administration Guide*.

Once your file system is created, it can be configured to mount whenever the GPFS daemon is started. This feature assures that whenever the system and disks are up, the file system will be available. When utilizing shared file system access among GPFS clusters, to reduce overall GPFS control traffic you may decide to mount the file system when it is first accessed. This is done through either the **mmremotefs** command or the **mmchfs** command using the **-A automount** option. GPFS mount traffic may be reduced by using automatic mounts instead of mounting at GPFS startup. Automatic mounts only produce additional control traffic at the point that the file system is first used by an application or user. Mounting at GPFS startup on the other hand produces additional control traffic at every GPFS startup. Thus startup of hundreds of nodes at once may be better served by using automatic mounts. However, when exporting the file system through Network File System (NFS) mounts, it might be useful to mount the file system when GPFS is started. For further information on shared file system access and the use of NFS with GPFS, see the *IBM Spectrum Scale: Administration and Programming Reference*.

Enhanced system flexibility

With GPFS, your system resources are not frozen. You can add or delete disks while the file system is mounted. When the time is favorable and system demand is low, you can rebalance the file system across all currently configured disks.

With the QoS capability, you can prevent I/O-intensive, long-running administration commands from dominating file system performance and significantly delaying other tasks. In addition, you can also add or delete nodes without having to stop and restart the GPFS daemon on all nodes.

Note: In the node quorum with tiebreaker disk configuration, GPFS has a limit of eight quorum nodes. If you add quorum nodes and exceed that limit, the GPFS daemon must be shutdown. Before you restart the daemon, switch quorum semantics to node quorum. For additional information, refer to "Quorum" on page 41.

In a SAN configuration where you have also defined NSD servers, if the physical connection to the disk is broken, GPFS dynamically switches disk access to the servers nodes and continues to provide data through NSD server nodes. GPFS falls back to local disk access when it has discovered the path has been repaired.

After GPFS has been configured for your system, depending on your applications, hardware, and workload, you can re-configure GPFS to increase throughput. You can set up your GPFS environment for your current applications and users, secure in the knowledge that you can expand in the future without jeopardizing your data. GPFS capacity can grow as your hardware expands.

Simplified storage management

GPFS provides storage management based on the definition and use of:

- Storage pools
- Policies
- Filesets

Storage pools

A *storage pool* is a collection of disks or RAIDs with similar properties that are managed together

as a group. Storage pools provide a method to partition storage within a file system. While you plan how to configure your storage, consider factors 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
- Improved reliability by providing for:
  - Replication based on need
  - Better failure containment

Policies

Files are assigned to a storage pool based on defined *policies*. Policies provide for:

Placement policies

Placing files in a specific storage pool when the files are created

File management policies

- Migrating files from one storage pool to another
- Deleting files based on file characteristics
- Changing the replication status of files
- Snapshot metadata scans and file list creation
- Compressing static files

Filesets

Filesets provide a method for partitioning a file system and allow administrative operations at a finer granularity than the entire file system. For example, filesets allow you to:

- Define data block and inode quotas at the fileset level
- Apply policy rules to specific filesets
- Create snapshots at the fileset level

For further information on storage pools, filesets, and policies, see the *IBM Spectrum Scale: Advanced Administration Guide*.

Simplified administration

GPFS offers many of the standard file system interfaces allowing most applications to execute without modification.

Operating system utilities are also supported by GPFS. That is, you can continue to use the commands you have always used for ordinary file operations (see Chapter 10, "Considerations for GPFS applications," on page 203). The only unique commands are those for administering the GPFS file system.

GPFS administration commands are similar in name and function to UNIX and Linux file system commands, with one important difference: *the GPFS commands operate on multiple nodes*. A single GPFS command can perform a file system function across the entire cluster. See the individual commands as documented in the *IBM Spectrum Scale: Administration and Programming Reference*.

GPFS commands save configuration and file system information in one or more files, collectively known as GPFS cluster configuration data files. The GPFS administration commands keep these files synchronized between each other and with the GPFS system files on each node in the cluster, thereby providing for accurate configuration data. See "Cluster configuration data files" on page 25.

The basic GPFS structure

GPFS is a clustered file system defined over one or more nodes. On each node in the cluster, GPFS consists of three basic components: administration commands, a kernel extension, and a multithreaded daemon.

For more information, see the following topics:

1. "GPFS administration commands"
2. "The GPFS kernel extension"
3. "The GPFS daemon"
4. For nodes in your cluster operating with the Linux operating system, "The GPFS open source portability layer" on page 7

For a detailed discussion of GPFS, see "GPFS architecture" on page 9.

GPFS administration commands

GPFS administration commands are scripts that control the operation and configuration of GPFS.

By default, GPFS commands can be executed from any node in the cluster. If tasks need to be done on another node in the cluster, the command automatically redirects the request to the appropriate node for execution. For administration commands to be able to operate, passwordless remote shell communications between the nodes is required.

For more information, see the *Requirements for administering a GPFS file system* topic in the *IBM Spectrum Scale: Administration and Programming Reference*.

The GPFS kernel extension

The GPFS kernel extension provides the interfaces to the operating system vnode and virtual file system (VFS) layer to register GPFS as a native file system.

Structurally, applications make file system calls to the operating system, which presents them to the GPFS file system kernel extension. GPFS uses the standard mechanism for the operating system. In this way, GPFS appears to applications as just another file system. The GPFS kernel extension will either satisfy these requests using resources which are already available in the system, or send a message to the GPFS daemon to complete the request.

The GPFS daemon

The GPFS daemon performs all I/O operations and buffer management for GPFS. This includes read-ahead for sequential reads and write-behind for all writes not specified as synchronous. I/O operations are protected by GPFS token management, which ensures consistency of data across all nodes in the cluster.

The daemon is a multithreaded process with some threads dedicated to specific functions. Dedicated threads for services requiring priority attention are not used for or blocked by routine work. In addition to managing local I/O, the daemon also communicates with instances of the daemon on other nodes to coordinate configuration changes, recovery, and parallel updates of the same data structures. Specific functions that execute in the daemon include:

1. Allocation of disk space to new files and newly extended files. This is done in coordination with the *file system manager*.
2. Management of directories including creation of new directories, insertion and removal of entries into existing directories, and searching of directories that require I/O.
3. Allocation of appropriate locks to protect the integrity of data and metadata. Locks affecting data that may be accessed from multiple nodes require interaction with the token management function.
4. Initiation of actual disk I/O on threads of the daemon.

5. Management of user security and quotas in conjunction with the file system manager.

The GPFS Network Shared Disk (NSD) component provides a method for cluster-wide disk naming and high-speed access to data for applications running on nodes that do not have direct access to the disks.

The NSDs in your cluster may be physically attached to all nodes or serve their data through an NSD server that provides a virtual connection. You are allowed to specify up to eight NSD servers for each NSD. If one server fails, the next server on the list takes control from the failed node.

For a given NSD, each of its NSD servers must have physical access to the same NSD. However, different servers can serve I/O to different non-intersecting sets of clients. The existing subnet functions in GPFS determine which NSD server should serve a particular GPFS client.

Note: GPFS assumes that nodes within a subnet are connected using high-speed networks. For additional information on subnet configuration, refer to "Using public and private IP addresses for GPFS nodes" on page 17.

GPFS determines if a node has physical or virtual connectivity to an underlying NSD through a sequence of commands that are invoked from the GPFS daemon. This determination, which is called *NSD discovery*, occurs at initial GPFS startup and whenever a file system is mounted.

Note: To manually cause this discovery action, use the **mmnsddiscover** command. For more information, see **mmnsddiscover command** in *IBM Spectrum Scale: Administration and Programming Reference*.

This is the default order of access used during NSD discovery:

1. Local block device interfaces for SAN, SCSI, IDE, or DASD disks
2. NSD servers

This order can be changed with the **useNSDserver** mount option.

It is suggested that you always define NSD servers for the disks. In a SAN configuration where NSD servers have also been defined, if the physical connection is broken, GPFS dynamically switches to the server nodes and continues to provide data. GPFS falls back to local disk access when it has discovered the path has been repaired. This is the default behavior, and it can be changed with the useNSDserver file system mount option.

For further information, see "Disk considerations" on page 51 and "NSD disk discovery" on page 24.

The GPFS open source portability layer

On Linux platforms, GPFS uses a loadable kernel module that enables the GPFS daemon to interact with the Linux kernel. Source code is provided for the portability layer so that the GPFS portability can be built and installed on a wide variety of Linux kernel versions and configuration.

When installing GPFS on Linux, you build a portability module based on your particular hardware platform and Linux distribution to enable communication between the Linux kernel and GPFS. For more information, see "Building the GPFS portability layer on Linux nodes" on page 149.

GPFS cluster configurations

A GPFS cluster can be configured in a variety of ways.

GPFS clusters can contain a mix of all supported node types including Linux, AIX, and Windows Server. These nodes can all be attached to a common set of SAN storage or through a mix of SAN and network attached nodes. Nodes can all be in a single cluster, or data can be shared across multiple clusters. A cluster can be contained in a single data center or spread across geographical locations. To determine which cluster configuration is best for your application start by determining the following:

- Application I/O performance and reliability requirements
- Properties of the underlying storage hardware
- Administration, security, and ownership considerations

Understanding these requirements helps you determine which nodes require direct access to the disks and which nodes should access the disks over a network connection through an NSD server.

There are four basic GPFS configurations:
- All nodes attached to a common set of LUNS
- Some nodes are NSD Clients
- A cluster is spread across multiple sites
- Data is shared between clusters

All nodes attached to a common set of LUNS

In this type of configuration, all of the nodes in the GPFS cluster are connected to a common set of LUNS (for example, over a SAN). The following are areas to consider with this configuration:
- The maximum number of nodes accessing a LUN you want to support
- The fact that you cannot mix different operating systems with GPFS to directly access the same set of LUNs on SAN.

For an example, see Figure 1.

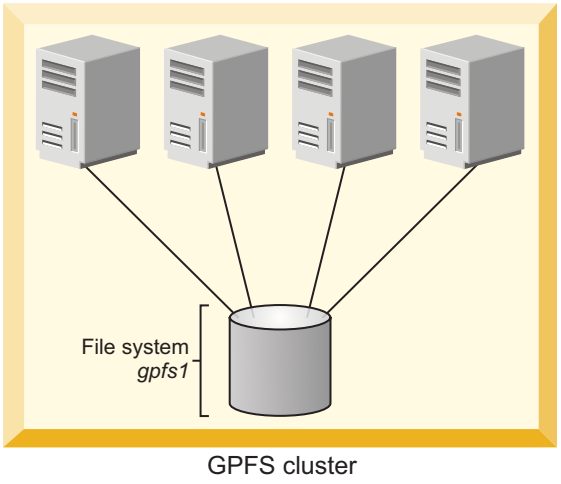


Figure 1. A cluster with disks that are SAN-attached to all nodes

GPFS Servers and GPFS clients

You can configure a GPFS cluster in which some nodes have a direct attachment to the disks and others access the disks through other GPFS nodes. This configuration is often used in large clusters or to provide a cost-effective, high-performance solution.

When a GPFS node is providing access to a disk for another GPFS node, it is called an NSD Server. The GPFS node accessing the data through an NSD server is called a GPFS client.

Sharing data across multiple GPFS clusters

GPFS allows you to share data across multiple GPFS clusters. Once a file system is mounted in another GPFS cluster all access to the data is the same as if you were in the host cluster. You can connect multiple

clusters within the same data center or across long distances over a WAN. In a multicluster configuration, each cluster can be placed in a separate administrative group simplifying administration or provide a common view of data across multiple organizations.

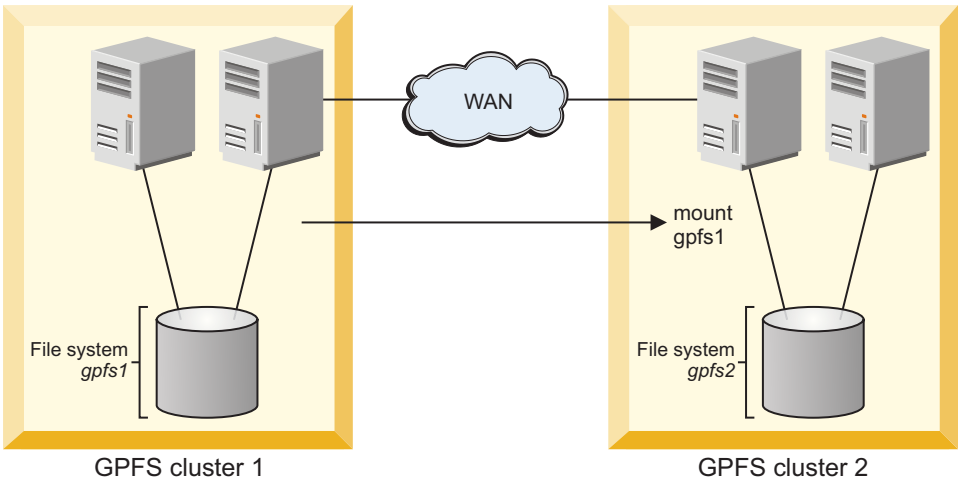


Figure 2. A multicluster configuration

Note: For more information, see the *Active file management* chapter in the *IBM Spectrum Scale: Advanced Administration Guide*.

GPFS architecture

Interaction between nodes at the file system level is limited to the locks and control flows required to maintain data and metadata integrity in the parallel environment.

A discussion of GPFS architecture includes:

- "Special management functions"
- "Use of disk storage and file structure within a GPFS file system" on page 12
- "GPFS and memory" on page 14
- "GPFS and network communication" on page 16
- "Application and user interaction with GPFS" on page 18
- "NSD disk discovery" on page 24
- "Failure recovery processing" on page 24
- "Cluster configuration data files" on page 25
- "GPFS backup data" on page 26

Special management functions

In general, GPFS performs the same functions on all nodes. It handles application requests on the node where the application exists. This provides maximum affinity of the data to the application.

There are three cases where one node provides a more global function affecting the operation of multiple nodes. These are nodes acting as:

1. "The GPFS cluster manager" on page 10
2. "The file system manager" on page 10
3. "The metanode" on page 11

The GPFS cluster manager

There is one GPFS cluster manager per cluster. The cluster manager is chosen through an election held among the set of quorum nodes designated for the cluster.

See "Quorum" on page 41 for more information.

The cluster manager performs the following tasks:

- Monitors disk leases
- Detects failures and manages recovery from node failure within the cluster.

  The cluster manager determines whether or not a quorum of nodes exists to allow the GPFS daemon to start and for file system usage to continue.
- Distributes certain configuration changes that must be known to nodes in remote clusters.
- Selects the *file system manager* node.

  The cluster manager prevents multiple nodes from assuming the role of file system manager, thereby avoiding data corruption, as the token management function resides on the file system manager node and possibly other nodes. For more information, see *Using multiple token servers* in *IBM Spectrum Scale: Advanced Administration Guide*.
- Handles UID mapping requests from remote cluster nodes.

To identify the cluster manager, issue the **mmlsmgr -c** command.

To change the cluster manager, issue the **mmchmgr -c** command.

The file system manager

There is one file system manager per file system, which handles all of the nodes using the file system.

The services provided by the file system manager include:

1. File system configuration

   Processes the following file system changes:

   - Adding disks
   - Changing disk availability
   - Repairing the file system

   Mount and unmount processing is performed on both the file system manager and the node requesting the service.
2. Management of disk space allocation

   Controls which regions of disks are allocated to each node, allowing effective parallel allocation of space.
3. Token management

   The file system manager node may also perform the duties of the token manager server. If you have explicitly designated some of the nodes in your cluster as file system manager nodes, then the token server load will be distributed among all of the designated manager nodes. For more information, see *Using multiple token servers* in *IBM Spectrum Scale: Advanced Administration Guide*.

   The token management server coordinates access to files on shared disks by granting tokens that convey the right to read or write the data or metadata of a file. This service ensures the consistency of the file system data and metadata when different nodes access the same file. The status of each token is held in two places:

   a. On the token management server

   b. On the token management client holding the token

   The first time a node accesses a file it must send a request to the token management server to obtain a corresponding **read** or **write** token. After having been granted the token, a node may continue to

read or write to the file without requiring additional interaction with the token management server. This continues until an application on another node attempts to read or write to the same region in the file.

The normal flow for a token is:

- A message to the token management server.

  The token management server then either returns a granted token or a list of the nodes that are holding conflicting tokens.
- The token management function at the requesting node then has the responsibility to communicate with all nodes holding a conflicting token and get them to relinquish the token.

  This relieves the token server of having to deal with all nodes holding conflicting tokens. In order for a node to relinquish a token, the daemon must give it up. First, the daemon must release any locks that are held using this token. This may involve waiting for I/O to complete.

4. Quota management

   In a quota-enabled file system, the file system manager node automatically assumes quota management responsibilities whenever the GPFS file system is mounted. Quota management involves:

   - Allocating disk blocks to nodes that are writing to the file system
   - Comparing the allocated space to the quota limits at regular intervals

   Notes:

   a. To reduce the number of space requests from nodes writing to the file system, the quota manager allocates more disk blocks than requested (see "Enabling quotas" on page 65). That allows nodes to write to the file system without having to go to the quota manager and check quota limits each time they write to the file system.
   b. File systems that have quotas enabled and more than 100,000 users or groups should avoid designating nodes as manager where memory is low or that are otherwise heavily loaded because of the high memory demands for quota manager operations.

The file system manager is selected by the cluster manager. If a file system manager should fail for any reason, a new file system manager is selected by the cluster manager and all functions continue without disruption, except for the time required to accomplish the takeover.

Depending on the application workload, the memory and CPU requirements for the services provided by the file system manager may make it undesirable to run a resource intensive application on the same node as the file system manager. GPFS allows you to control the pool of nodes from which the file system manager is chosen through:

- The **mmcrcluster** command, when creating your cluster
- The **mmaddnode** command, when adding nodes to your cluster
- The **mmchnode** command, to change a node's designation at any time

These preferences are honored except in certain failure situations where multiple failures occur. For more information, see *Multiple file system manager failures* in *IBM Spectrum Scale: Problem Determination Guide*. You may list which node is currently assigned as the file system manager by issuing the **mmlsmgr** command or change which node has been assigned to this task through the **mmchmgr** command.

The metanode

There is one metanode per open file. The metanode is responsible for maintaining file metadata integrity.

In almost all cases, the node that has had the file open for the longest continuous period of time is the metanode. All nodes accessing a file can read and write data directly, but updates to metadata are written only by the metanode. The metanode for each file is independent of that for any other file and can move to any node to meet application requirements.

CES node (protocol node)

Nodes in the cluster can be designated to be CES nodes using **mmchnode --ces-enable** *Node*. Only nodes that are designated as CES nodes can serve integrated protocol function. Each CES node will serve each of the protocols (NFS, SMB, Object) that are enabled. CES IP addresses assigned for protocol serving can failover to any of CES nodes that are up based on the failback policy configured. CES functionality can be designated only on nodes running Red Hat Enterprise Linux 7 or 7.1, and all the CES nodes must have the same platform (either all Intel or all POWER® Big Endian). For more information about Cluster Export Services, see *IBM Spectrum Scale: Advanced Administration Guide*.

Use of disk storage and file structure within a GPFS file system

A file system (or stripe group) consists of a set of disks that are used to store file metadata as well as data and structures used by GPFS, including quota files and GPFS recovery logs.

When a disk is assigned to a file system, a *file system descriptor* is written on each disk. The file system descriptor is written at a fixed position on each of the disks in the file system and is used by GPFS to identify this disk and its place in a file system. The file system descriptor contains file system specifications and information about the state of the file system.

Within each file system, files are written to disk as in other UNIX file systems, using inodes, indirect blocks, and data blocks. Inodes and indirect blocks are considered *metadata*, as distinguished from data, or actual file content. You can control which disks GPFS uses for storing metadata when you create the file system using the **mmcrfs** command or when modifying the file system at a later time by issuing the **mmchdisk** command.

Each file has an inode containing information such as file size, time of last modification, and extended attributes. The inodes of very small files and directories actually hold data. This increases performance for small files and directories. The inodes of small files also contain the addresses of all disk blocks that comprise the file data. A large file can use too many data blocks for an inode to directly address. In such a case, the inode points instead to one or more levels of indirect blocks that are deep enough to hold all of the data block addresses. This is the indirection level of the file.

The metadata for each file is stored in the inode and contains information such as file name, file size, and time of last modification. The inodes of small files also contain the addresses of all disk blocks that comprise the file data. When a file is large, it typically requires too many data blocks for an inode to directly address. In this case the inode points instead to one or more levels of indirect blocks. These trees of additional metadata space for a file can hold all of the data block addresses for very large files. The number of levels required to store the addresses of the data block is referred to as the indirection level of the file.

A file starts out with direct pointers to data blocks in the inode; this is considered a zero level of indirection. As the file increases in size to the point where the inode cannot hold enough direct pointers, the indirection level is increased by adding an indirect block and moving the direct pointers there. Subsequent levels of indirect blocks are added as the file grows. The dynamic nature of the indirect block structure allows file sizes to grow up to the file system size.

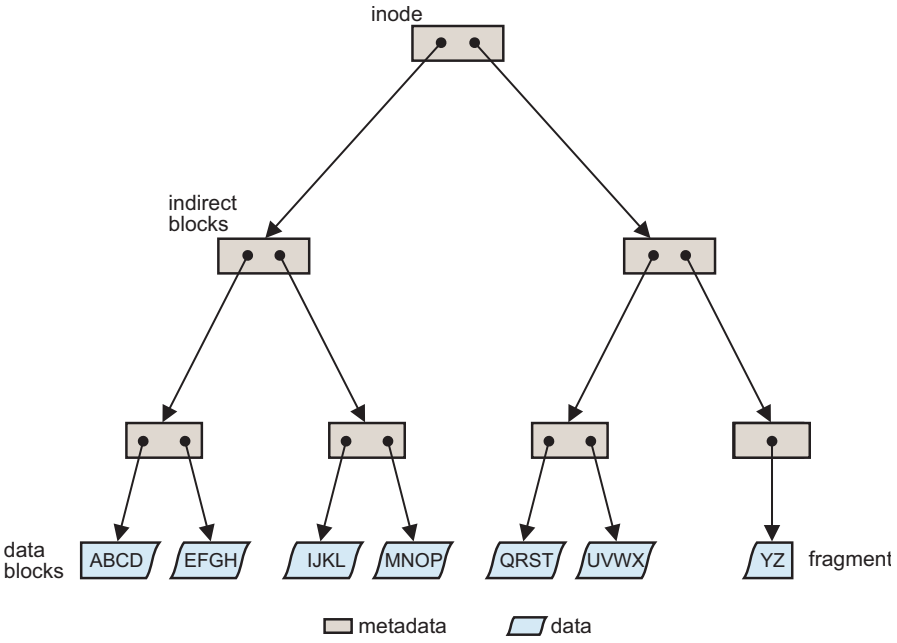


Figure 3. GPFS files have a typical UNIX structure

File system limitations:

1. The maximum number of mounted file systems within a GPFS cluster is 256.
2. The supported file system size depends on the version of GPFS that is installed.
3. The maximum number of files within a file system cannot exceed the architectural limit.

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

GPFS uses the file system descriptor to find all of the disks that make up the file system's stripe group, including their size and order. Once the file system descriptor is processed, it is possible to address any block in the file system. In particular, it is possible to find the first inode, which describes the *inode file*, and a small number of inodes that contain the rest of the file system information. The inode file is a collection of fixed length records that represent a single file, directory, or link. The unit of locking is the single inode. Specifically, there are fixed inodes within the inode file for the following:

- Root directory of the file system
- *Block allocation map*, which is a collection of bits that represent the availability of disk space within the disks of the file system. One unit in the allocation map represents a subblock or 1/32 of the block size of the file system. The allocation map is broken into regions that reside on disk sector boundaries. The number of regions is set at file system creation time by the parameter that specifies how many nodes will access this file system. The regions are separately locked and, as a result, different nodes can be allocating or de-allocating space represented by different regions independently and concurrently.
- *Inode allocation map*, which represents the availability of inodes within the inode file. The *Inode allocation map* is located in the *inode allocation file*, and represents all the files, directories, and links that can be created. The **mmchfs** command can be used to change the maximum number of files that can be created in the file system up to the architectural limit.

The data contents of each of these files are taken from the data space on the disks. These files are considered metadata and are allocated only on disks where metadata is allowed.

Quota files

For file systems with quotas enabled, quota files are created at file system creation time.

Note: Starting in GPFS 4.1, quota files no longer exist externally; therefore, use **mmbackupconfig -Q** to back up GPFS quota information.

There are three quota files for a file system:

- **user.quota** for users
- **group.quota** for groups
- **fileset.quota** for filesets

For every user who works within the file system, the **user.quota** file contains a record of limits and current usage within the file system for the individual user. If default quota limits for new users of a file system have been established, this file also contains a record for that value.

For every group whose users work within the file system, the **group.quota** file contains a record of common limits and the current usage within the file system of all the users in the group. If default quota limits for new groups of a file system have been established, this file also contains a record for that value.

For every fileset, the **fileset.quota** file contains a record of limits and current usage within the fileset. If default quota limits for filesets have been established, this file also contains a record for that value. The quota limit on blocks and inodes in a fileset are independent of the limits for specific users or groups of users. During allocation, the corresponding the limits for users, groups, and filesets are checked and the lowest threshold is applied.

Quota files are found through a pointer in the file system descriptor. Only the file system manager has access to the quota files. For backup purposes, quota files are also accessible as regular files in the root directory of the file system.

GPFS recovery logs

GPFS recovery logs are created at file system creation. Additional recovery logs are automatically created as needed. The file system manager assigns a recovery log to each node that accesses the file system.

Recovery logs are replicated only if default metadata replication is turned on (**-m 2**) or if explicitly enabled for logs (**--log-replicas 2**). You can check to see if log replication is enabled for a file system using the **mmlsfs** command and looking at the value of the **-m** and **--log-replicas** parameters. If both are set, the **--log-replicas** value takes precedence over the **-m** value for log replication.

GPFS maintains the atomicity of the on-disk structures of a file through a combination of rigid sequencing of operations and logging. The data structures maintained are the inode, indirect blocks, the allocation map, and the data blocks. Data blocks are written to disk before any control structure that references the data is written to disk. This ensures that the previous contents of a data block can never be seen in a new file. Allocation blocks, inodes, and indirect blocks are written and logged in such a way that there will never be a pointer to a block marked unallocated that is not recoverable from a log.

There are certain failure cases where blocks are marked allocated but not yet assigned to a file, and these can be recovered by running **mmfsck** in online or offline mode. Log recovery is run as part of:

1. The recovery of a node failure affecting the objects that the failed node might have had locked.
2. A **mount** after the file system has been unmounted everywhere.

Note: Space that is not used by metadata, quota files, and recovery logs is used for user data and directories and allocated from the block allocation map as needed.

GPFS and memory

GPFS uses three areas of memory: memory allocated from the kernel heap, memory allocated within the daemon segment, and shared segments accessed from both the daemon and the kernel.

Memory allocated from the kernel heap
GPFS uses kernel memory for control structures such as vnodes and related structures that establish the necessary relationship with the operating system.

Memory allocated within the daemon segment
GPFS uses daemon segment memory for file system manager functions. Because of that, the file system manager node requires more daemon memory since token states for the entire file system are initially stored there. File system manager functions requiring daemon memory include:

- Structures that persist for the execution of a command
- Structures that persist for I/O operations
- States related to other nodes

The file system manager is a token manager, and other nodes may assume token management responsibilities; therefore, any manager node may consume additional memory for token management. For more information, see *Using multiple token servers* in *IBM Spectrum Scale: Advanced Administration Guide*.

Shared segments accessed from both the daemon and the kernel
Shared segments consist of both pinned and unpinned memory that is allocated at daemon startup. The initial values are the system defaults. However, you can change these values later using the **mmchconfig** command. See "Cluster configuration file" on page 51.

The pinned memory is called the *pagepool* and is configured by setting the **pagepool** cluster configuration parameter. This pinned area of memory is used for storing file data and for optimizing the performance of various data access patterns. In a non-pinned area of the shared segment, GPFS keeps information about open and recently opened files. This information is held in two forms:

1. A full inode cache
2. A stat cache

Pinned memory

GPFS uses pinned memory (also called **pagepool** memory) for storing file data and metadata in support of I/O operations.

With some access patterns, increasing the amount of **pagepool** memory can increase I/O performance. Increased pagepool memory can be useful in the following cases:

- There are frequent writes that can be overlapped with application execution.
- There is frequent reuse of file data that can fit in the pagepool.
- The I/O pattern contains various sequential reads large enough that the prefetching data improves performance.

Pinned memory regions cannot be swapped out to disk, which means that GPFS will always consume at least the value of pagepool in system memory. So consider the memory requirements of GPFS and other applications running on the node when determining a value for **pagepool**.

Non-pinned memory

There are two levels of cache used to store file metadata.

Inode cache
The inode cache contains copies of inodes for open files and for some recently used files that are no longer open. The **maxFilesToCache** parameter controls the number of inodes cached by GPFS. Every open file on a node consumes a space in the inode cache. Additional space in the inode cache is used to store the inodes for recently used files in case another application needs that data.

The number of open files can exceed the value defined by the **maxFilesToCache** parameter to enable applications to operate. However, when the **maxFilesToCache** number is exceeded, there is not more caching of recently open files, and only open file inode data is kept in the cache.

Stat cache

The stat cache contains enough information to respond to inquiries about the file and open it, but not enough information to read from it or write to it. There is sufficient data from the inode in the stat cache to respond to a **stat()** call (for example, when issuing the **ls -l** command on a UNIX or Linux node). A stat cache entry consumes significantly less memory than a full inode. The default value stat cache is four times the **maxFilesToCache** parameter. This value may be changed through the **maxStatCache** parameter on the **mmchconfig** command. Stat cache entries are kept for the following:

- Recently accessed files
- Directories recently accessed by a number of **stat()** calls

Notes:

1. GPFS prefetches data for stat cache entries if a pattern of use indicates this will be productive (for example, if a number of **ls -l** commands issued for a large directory).
2. Each entry in the inode cache and the stat cache requires appropriate tokens:
   a. To ensure the cached information remains correct
   b. For the storage of tokens on the file system manager node
3. Depending on the usage pattern, system performance may degrade when an information update requires revoking a token. This happens when two or more nodes share the same information and the most recent information is moved to a different location. When the current node needs to access the updated information, the token manager must revoke the token from the current node before that node can access the information in the new location.

GPFS and network communication

Within the GPFS cluster, you can specify different networks for GPFS daemon communication and for GPFS command usage.

You make these selections using the **mmaddnode**, **mmchnode**, and **mmcrcluster** commands. In these commands, the node descriptor allows you to specify separate node interfaces for those functions on each node. The correct operation of GPFS is directly dependent upon these selections.

GPFS may not work properly if there is a firewall enabled on the nodes within the cluster. To ensure proper operation, you must either configure the firewall to allow the appropriate ports or disable the firewall. For more information, see *GPFS port usage* in *IBM Spectrum Scale: Advanced Administration Guide*.

GPFS daemon communication

In a cluster environment, the GPFS daemon depends on the correct operation of TCP/IP.

These dependencies exist because:

- The communication path between nodes must be built at the first attempt to communicate.
- Each node in the cluster is required to communicate with the cluster manager and the file system manager during startup and mount processing.
- Once a connection is established, it must remain active until the GPFS daemon is shut down on the nodes.

Note: Establishing other communication paths depends upon application usage among nodes.

The daemon also uses sockets to communicate with other instances of the file system on other nodes. Specifically, the daemon on each node communicates with the file system manager for allocation of logs, allocation segments, and quotas, as well as for various recovery and configuration flows. GPFS requires

an active internode communications path between all nodes in a cluster for locking, metadata coordination, administration commands, and other internal functions. The existence of this path is necessary for the correct operation of GPFS. The instance of the GPFS daemon on a node will go down if it senses that this communication is not available to it. If communication is not available to another node, one of the two nodes will exit GPFS.

Using public and private IP addresses for GPFS nodes:

GPFS permits the system administrator to set up a cluster such that both public and private IP addresses are in use. For example, if a cluster has an internal network connecting some of its nodes, it is advantageous to use private IP addresses to communicate on this network, and public IP addresses to communicate to resources outside of this network.

Public IP addresses are those that can be used to access the node from any other location for which a connection exists. Private IP addresses may be used only for communications between nodes directly connected to each other with a communications adapter. Private IP addresses are assigned to each node at hardware setup time, and must be in a specific address range (IP addresses on a 10.0.0.0, 172.16.0.0, or 192.168.0.0 subnet). For more information on private IP addresses, refer to RFC 1597 - Address Allocation for Private Internets (www.ip-doc.com/rfc/rfc1597).

The **subnets** operand on the **mmchconfig** command specifies an ordered list of **subnets** available to GPFS for private TCP/IP communications. Each subnet listed may have a list of cluster names (allowing shell-style wild cards) that specifies other GPFS clusters that have direct access to the same subnet.

When the GPFS daemon starts on a node, it obtains a list of its own IP addresses and associated subnet masks from its local IP configuration. For each IP address, GPFS checks whether that address is on one of the subnets specified on the **subnets** configuration parameter. It records the list of its matching IP addresses and subnet masks, and then listens for connections on any of these addresses. If any IP address for the node (specified when the cluster was created or when the node was added to the cluster), is not specified with the **subnets** configuration parameter, GPFS automatically adds it to the end of the node's IP address list.

Therefore, when using public IP addresses for a node, there is no need to explicitly list the public IP subnet with the **subnets** configuration parameter. For example, the normal way to configure a system would be to use host names that resolve to the external Ethernet IP address in the **mmcrcluster** command, and then, if the system administrator wants GPFS to use the High Performance Switch within the cluster, add one **subnets** configuration parameter for the HPS subnet. It is acceptable to add two **subnets** configuration parameters, one for the HPS and one for the external Ethernet, making sure that they are in that order. In this case it does not matter which of each node's two addresses was specified when the cluster was created or when the node was added to the cluster. For example, to add remote access to an existing cluster that was using only switch addresses, it is sufficient to add two **subnets** configuration parameters.

When a node joins a cluster (its own cluster on startup, or another cluster when mounting a file system owned by another cluster), the node sends its list of IP addresses (ordered according to the order of **subnets** configuration parameters) to the cluster manager node, which forwards the list to all other nodes as part of the join protocol. No other additional information needs to be propagated.

When a node attempts to establish a connection to another node, GPFS determines the destination IP address to use according to this procedure:

1. For each of its own IP addresses, it searches the other node's list of IP addresses for an address that is on the same subnet.
   - For normal public IP addresses this is done by comparing IP address values ANDed with the node's subnet mask for its IP address.

- For private IP addresses GPFS assumes that two IP addresses are on the same subnet only if the two nodes are within the same cluster, or if the other node is in one of the clusters explicitly listed in the **subnets** configuration parameter.

2. If the two nodes have more than one IP address pair on a common subnet, GPFS uses the first one found according to the order of **subnets** specified in the initiating node's configuration parameters.
3. If there are no two IP addresses on the same subnet, GPFS uses the last IP address in each node's IP address list. In other words, the last subnet specified in the **subnets** configuration parameter is assumed to be on a network that is accessible from the outside.

For more information and an example, see *Using remote access with public and private IP addresses* in *IBM Spectrum Scale: Advanced Administration Guide*.

Network communication and GPFS administration commands

Socket communications are used to process GPFS administration commands. Depending on the nature of the command, GPFS may process commands either on the node issuing the command or on the file system manager. The actual command processor merely assembles the input parameters and sends them along to the daemon on the local node using a socket.

Some GPFS commands permit you to specify a separate administrative network name. You make this specification using the **AdminNodeName** field of the node descriptor. For additional information, see *IBM Spectrum Scale: Administration and Programming Reference* for descriptions of these commands:

- **mmaddnode**
- **mmchnode**
- **mmcrcluster**

If the command changes the state of a file system or its configuration, the command is processed at the file system manager. The results of the change are sent to all nodes and the status of the command processing is returned to the node, and eventually, to the process issuing the command. For example, a command to add a disk to a file system originates on a user process and:

1. Is sent to the daemon and validated.
2. If acceptable, it is forwarded to the file system manager, which updates the file system descriptors.
3. All nodes that have this file system are notified of the need to refresh their cached copies of the file system descriptor.
4. The return code is forwarded to the originating daemon and then to the originating user process.

Be aware this chain of communication may allow faults related to the processing of a command to occur on nodes other than the node on which the command was issued.

Application and user interaction with GPFS

There are four ways to interact with a GPFS file system.

You can interact with a GPFS file system using:

- Operating system commands, which are run at GPFS daemon initialization time or at file system mount time (see "Operating system commands" on page 19)
- Operating system calls such as **open()**, from an application requiring access to a file controlled by GPFS (see "Operating system calls" on page 19)
- GPFS commands described in *IBM Spectrum Scale: Administration and Programming Reference* (see also "GPFS command processing" on page 23)
- GPFS programming interfaces described in *IBM Spectrum Scale: Administration and Programming Reference* and the *IBM Spectrum Scale: Data Management API Guide*

Operating system commands

Operating system commands operate on GPFS data during the following scenarios.

- The initialization of the GPFS daemon
- The mounting of a file system

Initialization of the GPFS daemon:

GPFS daemon initialization can be done automatically as part of the node startup sequence, or manually using the **mmstartup** command.

The daemon startup process loads the necessary kernel extensions, if they have not been previously loaded by another process. The daemon then waits for the cluster manager to declare that a quorum exists. When quorum is achieved, the cluster manager changes the state of the group from *initializing* to *active*. You can see the transition to active state when the "mmfsd ready" message appears in the GPFS log file (**/var/adm/ras/mmfs.log.latest**) or by running the **mmgetstate** command. When this state changes from *initializing* to *active*, the daemon is ready to accept mount requests.

The mounting of a file system:

GPFS file systems are mounted using the GPFS **mmmount** command.

On AIX or Linux you can also use the operating system's **mount** command. GPFS mount processing builds the structures required to provide a path to the data and is performed on both the node requesting the mount and the file system manager node. If there is no file system manager, a call is made to the cluster manager, which appoints one. The file system manager ensures that the file system is ready to be mounted. The file system manager ensures that there are no conflicting utilities being run by the **mmfsck** or **mmcheckquota** commands, that all of the disks are available, and that any necessary file system log processing is completed to ensure that metadata on the file system is consistent.

On the local node, the control structures required for a mounted file system are initialized and the token management function domains are created. In addition, paths to each of the disks that make up the file system are opened. Part of mount processing involves unfencing the disks, which may be necessary if this node had previously failed. This is done automatically without user intervention. If insufficient disks are up, the mount will fail. That is, in a replicated system if two disks are down in different failure groups, the mount will fail. In a non-replicated system, one disk down will cause the mount to fail.

Operating system calls

The most common interface to files residing in a GPFS file system is through normal file system calls to the operating system.

When a file is accessed, the operating system submits the request to the GPFS kernel extension, which attempts to satisfy the application request using data already in memory. If this can be accomplished, control is returned to the application through the operating system interface. If the data is not available in memory, the request is transferred for execution by a daemon thread. The daemon threads wait for work in a system call in the kernel, and are scheduled as necessary. Services available at the daemon level include the acquisition of tokens and disk I/O.

Operating system calls operate on GPFS data during:

- The opening of a file
- The reading of data
- The writing of data

Opening a GPFS file:

The **open** of a file residing in a GPFS file system involves the application making a call to the operating system specifying the name of the file.

Processing of a file **open** involves two stages:

1. Identifying the file specified by the application
2. Building the required data structures based on the inode

The kernel extension code processes the directory search. If the required information is not in memory, the daemon is called to acquire the necessary tokens for the directory or part of the directory needed to resolve the lookup, then reads the directory from disk into memory.

The lookup process occurs one directory at a time in response to calls from the operating system. In the final stage of **open**, the inode for the file is read from disk and connected to the operating system vnode structure. This requires acquiring locks on the inode and a lock that indicates the presence to the metanode. The metanode is discovered or created any time a file is opened.

- If no other node has this file open, this node becomes the metanode.
- If another node has a previous open, then that node is the metanode and this node interfaces directly with the metanode for metadata operations.

If the **open** involves the creation of a new file, the appropriate locks are obtained on the parent directory and the inode allocation file block. The directory entry is created, an inode is selected and initialized and then **open** processing is completed.

Reading file data:

The GPFS **read** function is invoked in response to a **read** system call.

File **read** processing falls into three levels of complexity based on system activity and status:

1. Buffers are available in memory
2. Tokens are available in memory but data must be read
3. Data and tokens must be acquired

At the completion of a **read**, a determination of the need for prefetching is made. GPFS computes a desired read-ahead for each open file based on the performance of the disks, the data access pattern and the rate at which the application is reading data. If additional prefetching is needed, a message is sent to the daemon that processes it asynchronously with the completion of the current **read**.

Buffer and locks available in memory:

The simplest **read** operation occurs when the data is already available in memory, either because it has been pre-fetched or because it has been read recently by another **read** call.

In either case, the buffer is locally locked and the data is copied to the application data area. The lock is released when the copy is complete. Note that no token communication is required because possession of the buffer implies that the node at least has a read token that includes the buffer. After the copying, prefetching is initiated if appropriate.

Tokens available locally but data must be read:

The second, more complex, type of **read** operation is necessary when the data is not in memory.

This occurs under three conditions:

1. The token has been acquired on a previous **read** that found no contention.
2. The buffer has been stolen for other uses.
3. On some random **read** operations.

   In the first of a series of reads, the token is not available locally, but in the second read it might be available.

In such situations, the buffer is not found and must be read from disk. No token activity has occurred because the node has a sufficiently strong token to lock the required region of the file locally. A message is sent to the daemon, which is handled on one of the waiting daemon threads. The daemon allocates a buffer, locks the file range that is required so the token cannot be stolen for the duration of the I/O, and initiates the I/O to the device holding the data. The originating thread waits for this to complete and is posted by the daemon upon completion.

Data and tokens must be acquired:

The third, and most complex **read** operation requires that tokens and data be acquired on the application node.

The kernel code determines that the data is not available locally and sends a message to the daemon waiting after posting the message. The daemon thread determines that it does not have the required tokens to perform the operation. In that case, a token acquire request is sent to the token management server. The requested token specifies a required length of that range of the file, which is needed for this buffer. If the file is being accessed sequentially, a desired range of data, starting at this point of this read and extending to the end of the file, is specified. In the event that no conflicts exist, the desired range is granted, eliminating the need for additional token calls on subsequent reads. After the minimum token needed is acquired, the flow proceeds as in step 3 on page 10 (token management).

Writing file data:

write processing is initiated by a system call to the operating system, which calls GPFS when the **write** involves data in a GPFS file system.

GPFS moves data from a user buffer into a file system buffer synchronously with the application **write** call, but defers the actual write to disk. This asynchronous I/O technique allows better scheduling of the disk I/O operations and improved performance. The file system buffers come from the memory allocated based on the **pagepool** parameter in the **mmchconfig** command. Increasing the size of the pagepool may allow more writes to be deferred, which can improve performance in certain workloads.

A block of data is scheduled to be written to a disk when:

- The application has specified a synchronous **write**.
- The system needs the memory used to buffer the written data.
- The file token has been revoked.
- The last byte of a block of a file being written sequentially is written.
- A system **sync** command is run.

Until one of these occurs, the data remains in GPFS memory.

write processing falls into three levels of complexity based on system activity and status:

1. Buffer available in memory
2. Tokens available locally but data must be read
3. Data and tokens must be acquired

Metadata changes are flushed under a subset of the same conditions. They can be written either directly, if this node is the metanode, or through the metanode, which merges changes from multiple nodes. This last case occurs most frequently if processes on multiple nodes are creating new data blocks in the same region of the file.

Buffer available in memory:

The simplest path involves a case where a buffer already exists for this block of the file but may not have a strong enough token.

This occurs if a previous **write** call accessed the block and it is still resident in memory. The write token already exists from the prior call. In this case, the data is copied from the application buffer to the GPFS buffer. If this is a sequential **write** and the last byte has been written, an asynchronous message is sent to the daemon to schedule the buffer for writing to disk. This operation occurs on the daemon thread overlapped with the execution of the application.

Token available locally but data must be read:

There are two situations in which the token may exist but the buffer does not.

1. The buffer has been recently stolen to satisfy other needs for buffer space.
2. A previous **write** obtained a desired range token for more than it needed.

In either case, the kernel extension determines that the buffer is not available, suspends the application thread, and sends a message to a daemon service thread requesting the buffer. If the **write** call is for a full file system block, an empty buffer is allocated since the entire block will be replaced. If the **write** call is for less than a full block and the rest of the block exists, the existing version of the block must be read and overlaid. If the **write** call creates a new block in the file, the daemon searches the allocation map for a block that is free and assigns it to the file. With both a buffer assigned and a block on the disk associated with the buffer, the **write** proceeds as it would in "Buffer available in memory."

Data and tokens must be acquired:

The third, and most complex path through **write** occurs when neither the buffer nor the token exists at the local node.

Prior to the allocation of a buffer, a token is acquired for the area of the file that is needed. As was true for **read**, if sequential operations are occurring, a token covering a larger range than is needed will be obtained if no conflicts exist. If necessary, the token management function will revoke the needed token from another node holding the token. Having acquired and locked the necessary token, the **write** will continue as in "Token available locally but data must be read."

The stat() system call:

The **stat()** system call returns data on the size and parameters associated with a file. The call is issued by the **ls -l** command and other similar functions.

The data required to satisfy the **stat()** system call is contained in the inode. GPFS processing of the **stat()** system call differs from other file systems in that it supports handling of **stat()** calls on all nodes without funneling the calls through a server.

This requires that GPFS obtain tokens that protect the accuracy of the metadata. In order to maximize parallelism, GPFS locks inodes individually and fetches individual inodes. In cases where a pattern can be detected, such as an attempt to **stat()** all of the files in a larger directory, inodes will be fetched in parallel in anticipation of their use.

Inodes are cached within GPFS in two forms:

1. Full inode
2. Limited stat cache form

The full inode is required to perform data I/O against the file.

The stat cache form is smaller than the full inode, but is sufficient to open the file and satisfy a **stat()** call. It is intended to aid functions such as **ls -l**, **du**, and certain backup programs that scan entire directories looking for modification times and file sizes.

These caches and the requirement for individual tokens on inodes are the reason why a second invocation of directory scanning applications may run faster than the first.

GPFS command processing

GPFS commands fall into two categories: those that are processed locally and those that are processed at the file system manager for the file system involved in the command.

The file system manager is used to process any command that alters the state of the file system. When commands are issued and the file system is not mounted, a file system manager is appointed for the task. The **mmchdisk** command and the **mmfsck** command represent two typical types of commands that are processed at the file system manager.

The mmchdisk command:

The **mmchdisk** command is issued when a failure that caused the unavailability of one or more disks has been corrected. The need for the command can be determined by the output of the **mmlsdisk** command.

mmchdisk performs four major functions:

- It changes the availability of the disk to **recovering**, and to **up** when all processing is complete. All GPFS utilities honor an availability of **down** and do not use the disk. **recovering** means that recovery has not been completed but the user has authorized use of the disk.
- It restores any replicas of data and metadata to their correct value. This involves scanning all metadata in the system and copying the latest to the recovering disk. Note that this involves scanning large amounts of data and potentially rewriting all data on the disk. This can take a long time for a large file system with a great deal of metadata to be scanned.
- It stops or suspends usage of a disk. This merely involves updating a disk state and should run quickly.
- Change disk attributes' metadata.

Subsequent invocations of **mmchdisk** will attempt to restore the replicated data on any disk left in with an availability of **recovering**

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

The mmfsck command:

The **mmfsck** command repairs file system structures.

The **mmfsck** command operates in two modes:

1. online
2. offline

For performance reasons, GPFS logging allows the condition where disk blocks are marked **used** but not actually part of a file after a node failure. The online version of **mmfsck** cleans up that condition.

Running **mmfsck -o -n** scans the file system to determine if correction might be useful. The online version of **mmfsck** runs on the file system manager and scans all inodes and indirect blocks looking for disk blocks that are allocated but not used. If authorized to repair the file system, it releases the blocks. If not authorized to repair the file system, it reports the condition to standard output on the invoking node.

The offline version of **mmfsck** is the last line of defense for a file system that cannot be used. It will most often be needed in the case where GPFS recovery logs are not available because of disk media failures. The **mmfsck** command runs on the file system manager and reports status to the invoking node. It is mutually incompatible with any other use of the file system and checks for any running commands or any nodes with the file system mounted. It exits if any are found. It also exits if any disks are **down** and require the use of **mmchdisk** to change them to **up** or **recovering**. The **mmfsck** command performs a full file system scan looking for metadata inconsistencies. This process can be lengthy on large file systems. It seeks permission from the user to repair any problems that are found, which may result in the removal of files or directories that are corrupt. The processing of this command is similar to those for other file systems.

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

NSD disk discovery

When the GPFS daemon starts on a node, it discovers the disks defined as NSDs by reading a disk descriptor that is written on each disk owned by GPFS. This enables the NSDs to be found regardless of the current operating system device name assigned to the disk.

On UNIX, NSD discovery is done by the GPFS shell script **/usr/lpp/mmfs/bin/mmdevdiscover**, which generates a list of available disk devices that appear in the node's local **/dev** file system. To override or enhance NSD discovery, you can create a script and name it **/var/mmfs/etc/nsddevices**. The user-created **nsddevices** script, if it exists, is executed before the default discovery process.

On Windows, NSDs have a GUID Partition Table (GPT) with a single GPFS partition. NSD discovery is done by scanning the system for a list of disks that contain a GPFS partition.

On all platforms, the list of disk devices is then used by the GPFS daemon to determine whether a device interface on the local node maps to an NSD name recorded in the configuration database. The process of mapping a device interface to an NSD involves GPFS opening each device in turn and reading any NSD volume identifier potentially recorded at sector two of the disk.

If GPFS discovers that an NSD volume identifier read from a disk device matches the volume identifier recorded with the NSD name in the GPFS configuration database, I/O for the local node proceeds over the local device interface.

If no device mapping appears on the local node for a particular NSD, I/O proceeds over the IP network to the first NSD server specified in the server list for that NSD. If the first NSD server in the server list is not available, I/O proceeds sequentially through the server list until it finds an available NSD server.

Consult the **/usr/lpp/mmfs/samples/nsddevices.sample** file for configuration guidelines on how to provide additional disk discovery capabilities unique to your configuration.

Failure recovery processing

In general, it is not necessary to understand the internals of GPFS failure recovery processing since it is done automatically. However, some familiarity with the concepts might be useful when failures are observed.

It should be noted that only one state change, such as the loss or initialization of a node, can be processed at a time and subsequent changes are queued. This means that the entire failure processing

must complete before the failed node can join the group again. All failures are processed first, which means that GPFS handles all failures prior to completing any recovery.

GPFS recovers from a node failure using join or leave processing messages that are sent explicitly by the cluster manager node. The cluster manager node observes that a node has failed when it no longer receives heartbeat messages from the node. The join or leave processing messages are broadcast to the entire group of nodes running GPFS, and each node updates its current status for the failing or joining node. Failure of the cluster manager node results in a new cluster manager being elected by the cluster. Then the newly elected cluster configuration manager node processes the failure message for the failed cluster manager.

When notified that a node has failed or that the GPFS daemon has failed on a node, GPFS invokes recovery for each of the file systems that were mounted on the failed node. If necessary, new file system managers are selected for any file systems that no longer have one.

The file system manager for each file system ensures the failed node no longer has access to the disks comprising the file system. If the file system manager is newly appointed as a result of this failure, it rebuilds token state by querying the other nodes in the group. After this is complete, the actual recovery of the log of the failed node proceeds. This recovery rebuilds the metadata that was being modified at the time of the failure to a consistent state. In some cases there may be blocks that are allocated that are not part of any file and are effectively lost until **mmfsck** is run, online or offline. After log recovery is complete, the locks held by the failed nodes are released for this file system. When this activity is completed for all file systems, failure processing is done. The last step of this process allows a failed node to rejoin the cluster.

Cluster configuration data files

GPFS commands store configuration and file system information in one or more files collectively known as GPFS cluster configuration data files. These files are not intended to be modified manually.

The GPFS administration commands are designed to keep these files synchronized between each other and with the GPFS system files on each node in the cluster. The GPFS commands constantly update the GPFS cluster configuration data files and any user modification made to this information may be lost without warning. On AIX nodes this includes the GPFS file system stanzas in **/etc/filesystems** and on Linux nodes the lists in **/etc/fstab**.

The GPFS cluster configuration data is stored in the **/var/mmfs/gen/mmsdrfs** file. This file is stored on the nodes designated as the *primary GPFS cluster configuration server* and, if specified, the secondary GPFS cluster configuration server. See "GPFS cluster configuration servers" on page 49. The first record in the **mmsdrfs** file contains a generation number. Whenever a GPFS command causes something to change in the cluster or any of the file systems, this change is reflected in the **mmsdrfs** file and the generation number is increased by one. The latest generation number is always recorded in the **mmsdrfs** file on the primary and secondary GPFS cluster configuration server nodes.

When running GPFS administration commands, it is necessary for the GPFS cluster configuration data to be accessible to the node running the command. Commands that update the **mmsdrfs** file require that both the primary and, if specified, the secondary GPFS cluster configuration server nodes are accessible. If one of the cluster configuration server nodes is inaccessible, you can designate a new primary or secondary cluster configuration servers using the **mmchcluster** command. Similarly, when the GPFS daemon starts up, at least one of the two server nodes must be accessible.

Starting with GPFS 4.1, the master copy of configuration data files may be stored redundantly on all quorum nodes instead of the separately designated primary/backup configuration server nodes. This method of storing configuration data is called the cluster configuration repository (CCR) and is the default for new clusters created on GPFS 4.1 or later. Existing clusters can be converted to the new repository type using the **–ccr-enable** option of the **mmchcluster** command.

Using CCR has the advantage that full read/write access to the configuration data remains available as long as a majority of quorum nodes are accessible. For example, in a cluster with five quorum nodes, commands that update the **mmsdrfs** file will continue to work normally, even if any two of the five quorum nodes have failed. In a two-node cluster with tiebreaker disks, it is still possible to run commands that change the **mmsdrfs** file if one of the two nodes has failed, as long as the surviving node has access to the tiebreaker disks. In general, full configuration command functionality remains available as long as enough nodes and, if specified, tiebreaker disks are accessible for GPFS to reach quorum. The CCR also has the advantage that it allows changing the tiebreaker disk configuration, including switching between node-based quorum and node quorum with tiebreaker, without first shutting down GPFS on all of the nodes.

Based on the information in the GPFS cluster configuration data, the GPFS commands generate and maintain a number of system files on each of the nodes in the GPFS cluster.

Linux **/etc/fstab**

On Linux nodes, contains lists for all GPFS file systems that exist in the cluster.

AIX **/etc/filesystems**

On AIX nodes, contains lists for all GPFS file systems that exist in the cluster.

All GPFS nodes

/var/mmfs/gen/mmfsNodeData

Contains GPFS cluster configuration data pertaining to the node.

/var/mmfs/gen/mmsdrfs

Contains a local copy of the **mmsdrfs** file found on the primary and secondary GPFS cluster configuration server nodes.

/var/mmfs/gen/mmfs.cfg

Contains GPFS daemon startup parameters.

GPFS backup data

The GPFS **mmbackup** command creates several files during command execution. Some of the files are temporary and deleted at the end of the backup operation. There are other files that remain in the root directory of the file system and should not be deleted.

The **mmbackup** command creates other files that begin with **.mmbackupShadow.***. These files are associated with the **mmbackup** command and are required for proper backups to be complete, so do not manually delete or change them.

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

Protocols support overview: Integration of protocol access methods with GPFS

Starting with V4.1.1, IBM Spectrum Scale provides additional protocol access methods in the Standard and Advanced editions of the product. Providing these additional file and object access methods and integrating them with GPFS offers several benefits. It enables users to consolidate various sources of data efficiently in one global namespace. It provides a unified data management solution and enables not just efficient space utilization but also avoids having to make unnecessary data moves just because access methods may be different.

The additional protocol access methods integrated with GPFS are file access using NFS and SMB and object access using OpenStack Swift. While each of these server functions (NFS, SMB and Object) uses

open source technologies, this integration adds value by providing the ability to scale and by providing high availability using the clustering technology in GPFS.

The integration of file and object serving with GPFS allows the ability to create NFS exports, SMB shares, and OpenStack Swift containers that have data in GPFS file systems for access by client systems that do not run GPFS. Some nodes (at least two recommended) in the GPFS cluster have to be designated as protocol nodes (also called CES nodes) from which (non-GPFS) clients can access data residing in and managed by GPFS using the appropriate protocol artifacts (exports/shares/containers). The protocol nodes need to have GPFS server license designations. The protocol nodes should be configured with "external" network addresses that will be used to access the protocol artifacts from clients. The (external) network addresses are different from the GPFS cluster address used to add the protocol nodes to the GPFS cluster. The integration provided allows the artifacts to be accessed from any of the protocol nodes via the configured network addresses. Further, the integration provided allows network addresses associated with protocol nodes to fail over to other protocol nodes when a protocol node fails. All the protocol nodes have to be running the Red Hat Enterprise Linux or the 12 operating system, and the protocol nodes must be all Power® (in big endian mode) or all Intel (although the other nodes in the GPFS cluster could be on other platforms and operating systems).

Note that like GPFS, the protocol serving functionality is also delivered (only) as software. The intent of the functionality is to provide access to data managed by GPFS via additional access methods. While the protocol function provides several aspect of NAS file serving, the delivery is not a NAS appliance. In other words, the GPFS-style command line interface requiring root access is still available, and therefore it is not like an appliance from an administrative management perspective. Role-based access control of the command line interface is not offered. Further, the type of workloads suited for this delivery continue to be those that require the scaling/consolidation aspects associated with traditional GPFS. It is important to note that some NAS workloads may not be suited for delivery in the current release (for instance, very extensive use of snapshots, or support for a very large number of SMB users). For more information, see IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html)

Along with the protocol-serving function, the delivery includes the **spectrumscale** installation toolkit as well as some performance monitoring infrastructure. The GPFS code, including the server function for the three (NFS, SMB, Object) protocols, along with the installation toolkit and performance monitoring infrastructure, are delivered via a self-extracting archive package (just like traditional GPFS). The use of the protocol server function requires additional licenses that need to be accepted. A GPFS package without protocols continues to be provided for those users who do not wish to accept these additional license terms. Note that even though some of the components provided are open source, the specific packages provided should be used. If there are existing versions of these open source packages on your system, they should be removed before installing our software.

Several new commands have been introduced to enable the use of the function described in the preceding sections. The new commands are **spectrumscale**, **mmces**, **mmuserauth**, **mmnfs**, **mmsmb**, **mmobj**, and **mmperfmon**. In addition, **mmdumpperfdata** and **mmprotocoltrace** have been provided to help with data collection and tracing. Existing GPFS commands that have been expanded with some options for protocols include **mmlscluster**, **mmchnode**, and **mmchconfig**. Further, **gpfs.snap** has been extended to include data gathering about the protocols to help with problem determination.

IBM Spectrum Scale 4.1.1 adds cluster export services (CES) infrastructure to support the integration of the NFS, SMB, and object servers. The NFS Server supports NFS v3 and the mandatory features in NFS v4.0. The SMB server support SMB 2, SMB 2.1, and the mandatory features of SMB 3.0. The object server supports the Kilo release of Openstack Swift along with Keystone v3. The CES infrastructure is responsible for (a) managing the setup for high-availability clustering used by the protocols; (b) monitoring the health of these protocols on the protocol nodes and raising events/alerts in the event of failures; (c) managing the addresses used for accessing these protocols including failover and failback of

these addresses because of protocol node failures. For information on the use of CES including administering and managing the protocols, see the *Implementing Cluster Export Services* chapter of *IBM Spectrum Scale: Advanced Administration Guide*.

IBM Spectrum Scale enables you to build a **data ocean** solution to eliminate silos, improve infrastructure utilization, and automate data migration to the best location or tier of storage anywhere in the world. You can start small with just a few commodity servers fronting commodity storage devices and then grow to a data lake architecture or even an ocean of data. IBM Spectrum Scale is a proven solution in some of the most demanding environments with massive storage capacity under the single global namespace. Furthermore, your data ocean can store either files or objects and you can run analytics on the data in-place, which means that there is no need to copy the data to run your jobs.

The **spectrumscale** installation toolkit is provided to help with the installation and configuration of GPFS as well as protocols. While it was designed for a user who may not be familiar with GPFS, it can help ease the installation and configuration process of protocols even for experienced GPFS administrators.

Note:

The installation toolkit can help with prechecks to validate environment, distribution of the RPMs from one node to the other nodes, and multiple GPFS administrative steps. **spectrumscale deploy** can be used to configure protocols on an existing GPFS cluster with an existing GPFS file system.

In addition to the installation toolkit, IBM Spectrum Scale 4.1.1 and later also adds a performance monitoring toolkit. Sensors to collect performance information are installed on all protocol nodes, and one of these nodes is designated as a collector node. The **mmperfmon query** command can be used to view the performance counters that have been collected. .

Some protocol use considerations:

- At time of release, several features in GPFS have not been explicitly tested with protocol functionality. These include Local Read Only Cache, Multicluster, Encryption, File Placement Optimizer, and Hierarchical Storage Management. This is expected to work with protocols and will be tested with protocols over time. However, if you use one of these features before IBM claims support of it, ensure that it is tested with the expected workloads before putting it in production.
- Use of Clustered NFS (CNFS) is not compatible with use of Clustered Export Service (CES). You must choose one or the other. CNFS (which uses kernel NFS, a different NFS stack than that used by the CES infrastructure) continues to be supported; however, if you choose CNFS, you cannot take advantage of the integration of SMB and Object server functionality. Note that if you choose to migrate from CNFS to CES, the CES infrastructure does not support the equivalent of CNFS group feature to control failover of IP addresses.
- Protocol nodes cannot be used to serve remote mounted file systems.
- For information regarding specific limitations about protocols and their integration with GPFS, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) and IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html).

NFS support overview

The NFS support for IBM Spectrum Scale™ enables clients to access the GPFS™ file system by using NFS clients with their inherent NFS semantics.

The following features are provided:

Clustered NFS Support

NFS clients can connect to any of the protocol nodes and get access to the exports defined. A clustered registry makes sure that all nodes see the same configuration data. This means that it does not matter to the client to which of the CES nodes the connections are established. Moreover, the state of opened files is also shared among the CES nodes so that data integrity is maintained. On failures, clients can reconnect to another cluster node as the IP addresses of failing nodes are automatically transferred to another healthy cluster node. The supported protocol levels are NFS version 3 (NFSv3) and NFS version 4 (NFSv4.0).

Export management commands

With the **mmnfs export** command, IBM Spectrum Scale provides a comprehensive entry point to manage all NFS-related export tasks such as creating, changing, and deleting NFS exports. The **mmnfs export** command follows the notions of supported options, that is, a limited set of NFS-related options that have proven useful.

Export configuration commands

With the **mmnfs configuration** command, IBM Spectrum Scale provides a tool for administering the global NFS configuration. You can use this command to set and list default settings such as the port number for the NFS service, the default access mode for exported file systems, and the NFS server log level. This command follows the notions of supported options, that is, a limited set of NFS-related options that have proven useful.

NFS monitoring

The monitoring framework detects issues with the NFS services and triggers failover in case of an unrecoverable error. Moreover, the **mmces** command provides a quick access to current and past system states and these states enable you to diagnose issues with the NFS services on the CES nodes. Issues that are detected and causing failover are, for example, GPFS daemon failures, node failures, or NFS service failures.

Integrated install

The integrated installer allows the installation of NFS services, CES framework, and the other protocols (SMB and Object), if desired.

NFS performance metrics

The NFS services provide performance metrics that are collected by the performance monitor framework. The **mmperfmon query** tool provides access to the most important NFS metrics through predefined queries.

Cross-protocol integration with SMB

IBM Spectrum Scale enables concurrent access to the same file data by using NFS, SMB, and native POSIX access (limitations apply). The first release does not allow the sharing of data through the traditional file protocols and Object.

Authentication and ID mapping

You can configure NFS services to authenticate against the most popular authentication services such as Microsoft Active Directory and LDAP. Mapping Microsoft security identifiers (SIDs) to the POSIX user and group IDs on the file server can either be done automatically or by using the external ID mapping service like RFC 2307. If none of the offered authentication and mapping schemes match the environmental requirements, the option to establish a user-defined configuration is available. The

mmuserauth service create command can be used to set up all authentication-related settings.

SMB support overview

The SMB support for IBM Spectrum Scale 4.1.1 and later allows clients to access the GPFS file system using SMB clients with their 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, that is, for the client it does not matter to which of the CES nodes the connections are established. Moreover, the state of opened files (share modes, open modes, access masks, locks, etc.) is also shared among the CES nodes so that data integrity is maintained. On failures, clients can reconnect to another cluster node as the IP addresses of failing nodes are transferred to another healthy cluster node. The supported protocol levels are SMB2 and the base functionality 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 and administering the global configuration. The **mmsmb** command follows the notions of supported options, that is, a limited set of SMB related options that have proven useful. Moreover, the Microsoft Management Console can be used to administer SMB exports.

- SMB monitoring

  The monitoring framework will detect issues with the SMB services and will trigger failover in case of an unrecoverable error. Moreover, the **mmces** command provide a quick access to current and past system states and aid to diagnose issues with the SMB services on the CES nodes. Issues detected and causing failover are, for example, GPFS daemon failures, node failures, or SMB service failures.

- Integrated install

  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. Thus not just the current metrics can be retrieved but also historic data is available (at some lower granularity). The two sets of metrics are firstly global SMB metrics (like number of connects and disconnects) and secondly 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 meta-data base CTDB are collected and exposed via the **mmperfmon query** command.

- Cross-protocol integration with NFS

  IBM Spectrum Scale 4.1.1 and later allows concurrent access to the same file data via SMB, NFS and native POSIX access (limitations apply). The first release does not allow to share data via the traditional file protocols and Object.

- Authentication and ID Mapping

  The SMB services can be configured to authenticate against the most popular 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 using the automatic or external ID mapping service like RFC 2307. If none of the offered authentication and mapping schemes matches the environmental requirements the option to establish a user-defined configuration is available. The **mmuserauth service create** command can be used to set up all authentication related settings.

Object storage support overview

IBM Spectrum Scale for object storage combines the benefits of IBM Spectrum Scale with the most widely used open source object store, OpenStack Swift.

Data centers are currently struggling to efficiently and cost-effectively store and manage vast amounts of data. The increasing number of application domains, such as analytics, online transaction processing (OLTP), and high-performance computing (HPC), have created silos of storage within data centers. With each new application, a new storage system can be required, forcing system administrators to become experts in numerous storage management tools.

In addition, the set of applications that includes mobile and web-based applications, archiving, backup, and cloud storage has recently created yet another type of storage system for the system administrator to manage: object storage. Objects cannot be updated after they are created (although they can be replaced, versioned, or removed), and in many cases the objects are accessible in an eventually consistent manner. These types of semantics are well suited for images, videos, text documents, virtual machine (VM) images, and other similar files.

IBM Spectrum Scale for object storage combines the benefits of IBM Spectrum Scale with the most widely used open source object store today, OpenStack Swift. In IBM Spectrum Scale for object storage, data is managed as objects and it can be accessed over the network by using RESTful HTTP-based APIs. This object storage system uses a distributed architecture with no central point of control, providing greater scalability and redundancy. IBM Spectrum Scale for object storage organizes data in the following hierarchy:

1. **Account**

   An account is used to group or isolate resources. Each object user is part of an account. Object users are mapped to an account and it can access only the objects that reside within the account. Each user needs to be defined with the set of user rights and privileges to perform a specific set of operations on the resources of the account to which it belongs. Users can be part of multiple accounts but it is mandatory that a user must be associated with at least one account. You must create at least one account before adding users. Account contains a list of containers in the object storage. You can also define quota at the account level.

2. **Container**

   Container contains objects and it lists object in the specified container. It provides a namespace for the objects. You can create any number of containers within an account.

3. **Object**

   Objects store data. You can create, modify, and delete objects. Accounts have containers, and containers store objects. Containers logically reside within the accounts. So, a container named "Documents" in two different storage accounts are two distinct containers within the cluster. Each object is accessed through a unique URL. The URLs in the API contain the storage account, container name, and object name. You can define quota both at the account and container levels.

IBM Spectrum Scale for object storage also offers the following features.

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. For more information about storage policies, see OpenStack Documentation for Storage Policies. IBM Spectrum Scale enhances storage policies to add the following functionality for object storage:

- **`file-access`** (unified file and object access)
- **`compression`**

You can use the **mmobj policy create** command to create a storage policy with the desired functionality from the available options. After a storage policy is created, you can specify that storage policy while creating new containers to associate that storage policy with those containers.

A fileset is associated the new storage policy. The name of the fileset can be provided optionally as an argument of the **mmobj policy create** command.For information on mapping of storage policy and fileset, see *Mapping of storage policies to filesets* in *IBM Spectrum Scale: Administration and Programming Reference*.

For information on creating, listing, and changing storage policies, see *Administering storage policies for object storage* and *mmobj command* in *IBM Spectrum Scale: Administration and Programming Reference*.

Unified file and object access overview

Unified file and object access allows use cases where you can access data using object as well as file interfaces.

Some of the key unified file and object access use cases are as follows:

- Accessing object using file interfaces and accessing file using object interfaces helps legacy applications designed for file to start integrating into the object world after data migration.
- It allows storage cloud data which is in form of objects to be accessed using files from applications designed to process files.
- It allows files exported using NFS or SMB, or files available on POSIX, to be accessible as objects using http to the end clients. This enables easy availability of file data on mobile devices such as smart phones or tablets which are more suited to REST based interfaces.
- Multi protocol access for file and object in the same namespace allows supporting and hosting **data oceans** of different types with multiple access options.

  For information about **data oceans**, see "Protocols support overview: Integration of protocol access methods with GPFS" on page 26.
- There is a rich set of placement policies for files (using **mmapplypolicy**) available with IBM Spectrum Scale. With unified file and object access, those placement policies can be leveraged for object data.
- Object stores are suitable for storing large amounts of data because they are highly scalable and they are an economical storage solution. To analyze large amounts of data, advanced analytics systems are used. However, porting the data from an object store to a distributed file system that the analytics system requires is complex and time intensive. For these scenarios, there is a need to access the object data using file interface so that analytics systems can use it.

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. Unified file and object access is deployed as an object storage policy. Unified file and object access provides the following capabilities to users:

- Ingest data through the object interface and access this data from the file interface
- Ingest data through the file interface and access this data from the object interface
- Ingest and access same data though object and file interfaces concurrently
- Manage authentication and authorization in unified file and object access

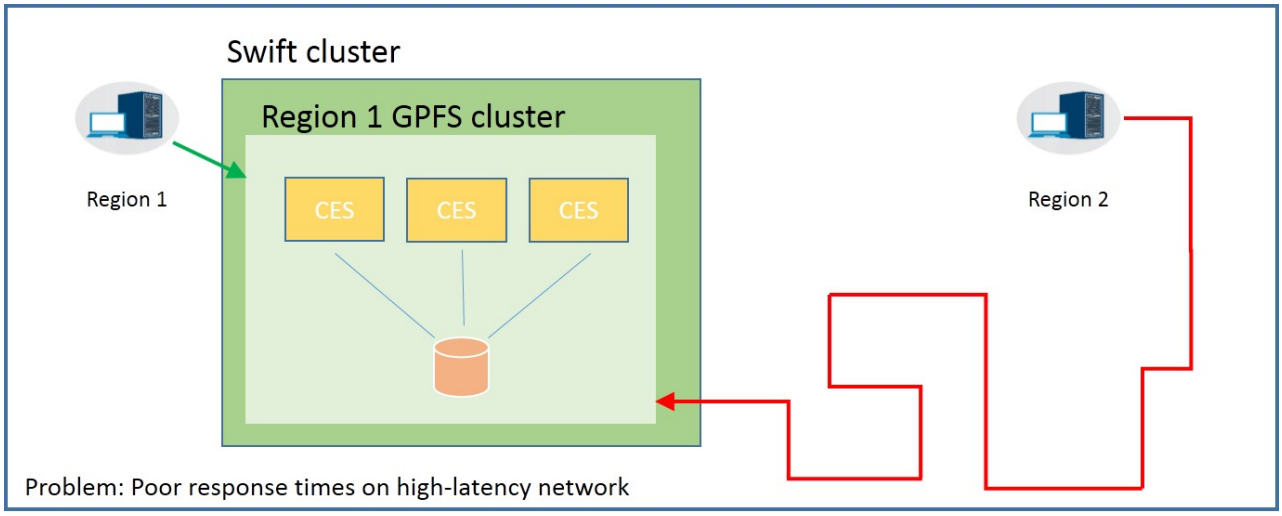
For more information, see *Unified file and object access in IBM Spectrum Scale* in *IBM Spectrum Scale: Administration and Programming Reference*.

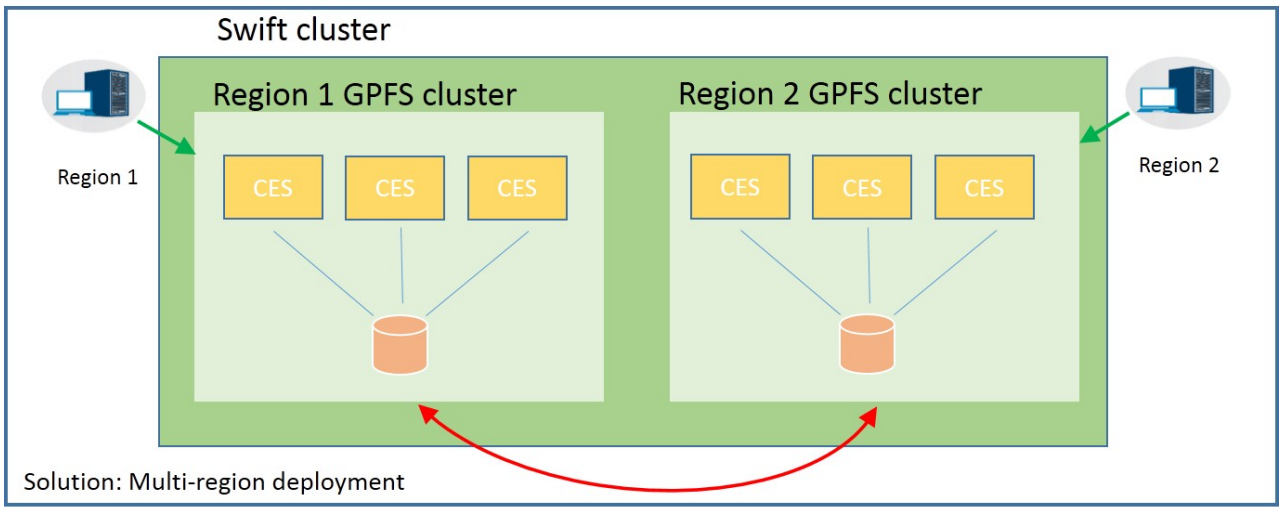
One of the key advantages of unified file and object access is the placement and naming of objects when stored on the file system. For more information, see *File path in unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

Overview of 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.

With multi-region support in Swift, the replication servers asynchronously copy data between remote clusters to keep them in sync. When the client uploads or downloads objects, the proxy server it connects to initially uses the devices in the same region as the proxy. If the proxy cannot find the data it needs in its own region, it then uses the other regions. In this way, the clients generally get fast access for the data in the close cluster and only be affected by the high-latency penalty when accessing data which has not yet been replicated.





Multi-region object deployment involves creating up to 3 independent CES clusters. Each CES cluster is a single region. The Swift environment and ring files are configured to map each cluster to an associated region index. The configuration is then synced manually between the clusters to enable the active-active replication.

Enabling multi-region support converts the underlying primary Swift account, container, and object rings to include all defined regions. By default, all data is stored in all regions. To store data in just a subset of the regions, storage policies can be used to create object rings which just store the objects in a subset of the regions. For more information on creating custom storage policies, see *mmobj command* in *IBM Spectrum Scale: Administration and Programming Reference*.

Only the first cluster can switch to multi-cluster after installation. Subsequent clusters need to be installed as multi-cluster environments due to the need for region numbers and storage policy indices to be globally consistent across clusters.

For information on planning a multi-region object deployment, see "Planning for multi-region object deployment" on page 97.

For information on enabling multi-region object deployment, see "Enabling multi-region object deployment initially" on page 133.

For information on adding a new region to a multi-region object deployment environment, see *Adding a region in a multi-region object deployment* in *IBM Spectrum Scale: Administration and Programming Reference*.

S3 API emulation

IBM Spectrum Scale uses Swift3 Middleware for OpenStack Swift, allowing access to IBM Spectrum Scale using the Amazon Simple Storage Service (S3) API. IBM Spectrum Scale for object storage includes S3 API emulation as an optional feature.

S3 API emulation can be either enabled during protocol deployment, initial object configuration, or later on.

- For information on enabling S3 API emulation during protocol deployment using the `-s3` option of the **`spectumscale config object`** command, see "Deploying protocols" on page 121.
- For information on enabling S3 API emulation during initial object configuration using the `--enable-s3` option of the **`mmobj swift base`** command, see *Configuring and enabling the Object protocol service* in *IBM Spectrum Scale: Administration and Programming Reference*.
- For information on enabling S3 API emulation if it is not enabled as part of the object base configuration, see *Changing the object base configuration to enable S3 API emulation* in *IBM Spectrum Scale: Administration and Programming Reference*.

Accessing the object storage through swift requests is not affected by enabling the S3 API emulation. When the S3 API emulation is enabled, the object service also recognizes S3 requests sent to the TCP port used by the object service (8080).

For more information on the S3 API, see the Amazon S3 documentation.

For limitations of the S3 API emulation support with IBM Spectrum Scale, see *Managing OpenStack access control lists using S3 API emulation* in *IBM Spectrum Scale: Administration and Programming Reference*.

Object capabilities

Object capabilities describe the object protocol features that are configured in the IBM Spectrum Scale cluster.

The following capabilities can be viewed:

- file-access (Unified file and object access)
- multi-region (Multi-region object deployment)
- s3 (Amazon S3 API emulation)

If unified file and object access has already been configured, changing the file-access capability can be used to enable or disable the related services. The other capabilities are for information only and must not be modified.

To use storage policies with the unified file and object access functionality enabled, you must enable the file-access capability first. Otherwise, the storage policy creation command (**mmobj policy create**) fails.

The **ibmobjectizer** and the **object-server-sof.conf** services are started only if the **file-access** capability is enabled. Disabling the **file-access** capability stops these services.

For information about enabling, disabling, and listing object capabilities, see *Managing object capabilities* in *IBM Spectrum Scale: Administration and Programming Reference*.

Cluster Export Services overview

Cluster Export Services (CES) includes support for monitoring high availability through protocols and commands.

High availability

With GPFS, you can configure a subset of nodes in the cluster to provide a highly available solution for exporting GPFS file systems by using the Network File System (NFS), Server Message Block (SMB), and Object protocols. The participating nodes are designated as Cluster Export Services (CES) nodes or protocol nodes. The set of CES nodes is frequently referred to as the *CES cluster*.

A set of IP addresses, the *CES address pool*, is defined and distributed among the CES nodes. As nodes enter and leave the GPFS cluster, the addresses in the pool can be redistributed among the CES nodes to provide high availability. Remote clients can use these addresses to access the cluster.

Monitoring

Each CES node runs a separate GPFS utility that monitors the state of the node. This utility includes checks of the CES addresses that are assigned to the node and a check of the processes that implement the enabled services in the CES cluster. Upon failure detection, the monitoring utility might mark the node as temporarily unable to participate in the CES cluster and reassign any addresses that are assigned to the node.

Protocol support

CES supports three export protocols: NFS, SMB, and Object. Each protocol can be enabled or disabled in the cluster. If a protocol is enabled in the CES cluster, all CES nodes run servers for that protocol.

Protocols are enabled and disabled with the **mmces** command:

mmces service enable nfs
: Enables the NFS protocol in the CES cluster.

mmces service disable obj
: Disables the Object protocol in the CES cluster.

Commands

To set or configure CES options, the following commands are used:

mmces
: Sets the CES address pool and other CES cluster configuration options.

mmnfs
Sets the NFS configuration operations.

mmobj
Sets the Object configuration operations.

mmsmb
Sets the SMB configuration operations.

mmuserauth
Configures the authentication methods that are used by the protocols.

For more information, see *mmces command*, *mmnfs command*, *mmobj command*, *mmsmb command*, and *mmuserauth command* in *IBM Spectrum Scale: Administration and Programming Reference*.

Restrictions

Cluster Export Services is limited to Linux nodes that are running Red Hat Enterprise Linux 7 (RHEL 7). 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).

If the SMB protocol is enabled, the CES cluster is limited to 16 nodes that must be of the same architecture.

Each CES node must be able to host all addresses in the CES address pool.

Each CES node must have network adapters capable of supporting all IP addresses in the CES address pool. The primary address of these adapters should not be used as a CES address.

Introduction to IBM Spectrum Scale GUI

The IBM Spectrum Scale management GUI provides an easy way to configure and manage various features that are available with the IBM Spectrum Scale system.

You can perform the following important tasks through the IBM Spectrum Scale 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

The following table provides an overview of the features associated with each GUI page.

Table 3. Features associated with IBM Spectrum Scale GUI pages

GUI Page	Function
Monitoring > **Dashboards**	Provides a dashboard to display the predefined performance charts and the customized charts that are marked as favorite charts in the Performance page.

Table 3. Features associated with IBM Spectrum Scale GUI pages (continued)

GUI Page	Function
Monitoring > Topology	Monitor the status of nodes and NSDs that are configured in the system.
Monitoring > Events	Monitor and troubleshoot the issues that are reported in the system.
Monitoring > Performance	Provides a list of pre-defined performance charts to monitor the performance of the system based on various aspects.
Monitoring > Capacity	Facilitates to monitor the capacity details that are reported at various levels.
Files > File Systems	View and manage file systems. File system creation through GUI is not supported in 4.2 release.
Files > Filesets	Create, view, and manage filesets.
Files > Snapshots	Create snapshots or snapshot rules to automate snapshot creation and retention.
Files > Quotas	Create and manage default, user, group, and fileset quotas at the file system level.
Files > Information Lifecycle	Create, manage, and delete policies that manage automated tiered storage of information.
Protocols > NFS Exports	Create and manage NFS exports. Protocols pages are displayed in the GUI only when the protocol feature is enabled on the cluster.
Protocols > SMB Shares	Create and manage SMB shares. Protocols pages are displayed in the GUI only when the protocol feature is enabled on the cluster.
Object > Accounts	Create and manage accounts and containers in the object storage. Object pages are displayed in the GUI only when the object feature is enabled on the cluster.
Object > Containers	Manage containers in object storage.
Object > Users	Create object users.
Object > Roles	Define roles for the object users.
Access > GUI Users	Create users and groups.
Access > File System ACL	Define ACL templates and apply ACLs on files and directories.
Settings > Event Notifications	Configure event notifications to notify administrators when events occur in the system.
Settings > Download Logs	Download logs to troubleshoot the issues that are reported in the system.
Settings > NFS Service	Specify NFS server settings and start or stop NFS services.
Settings > SMB Service	Specify SMB server settings and start or stop SMB services.
Settings > Object Administrator	Define object administrator who can manage accounts in the object storage.

Table 3. Features associated with IBM Spectrum Scale GUI pages (continued)

GUI Page	Function
Settings > **GUI Preferences**	Specify a message that can be displayed in the login page.

Note: The default user name and password to access the IBM Spectrum Scale management GUI are `admin` and `admin001` respectively.

Assistance for understanding the features associated with a GUI page

The following three levels of assistance are available for the GUI users:

1. **Hover help:** When you hover the mouse over the tiny question mark that is placed next to the field label, the system displays a brief description of the feature that is associated with that field. Hover help is only available for the important and complex fields.
2. **Context-sensitive help:** Provides a detailed explanation of the features that are associated with the page. The context-sensitive help files are available in the help menu, which is placed in the upper right corner of the GUI page.
3. **Knowledge Center:** This is the third level of information where the users find entire details of the product. Link to the IBM Spectrum Scale Knowledge Center is also available in the help menu, which is placed in the upper right corner of the GUI page.

Related concepts:

"Manually installing IBM Spectrum Scale management GUI" on page 144
The management GUI provides an easy way for the users to configure, manage, and monitor the IBM Spectrum Scale system.

Chapter 2. Planning for IBM Spectrum Scale

Planning for GPFS

Although you can modify your GPFS configuration after it has been set, a little consideration before installation and initial setup will reward you with a more efficient and immediately useful file system.

During configuration, GPFS requires you to specify several operational parameters that reflect your hardware resources and operating environment. During file system creation, you can specify parameters that are based on the expected size of the files or you can let the default values take effect.

The **spectrumscale** installation toolkit is also available to assist with GPFS installation on Linux nodes. For more information, see "Overview of the spectrumscale installation toolkit" on page 106

Planning for GPFS includes:

- "Hardware requirements"
- "Software requirements" on page 40
- "IBM Spectrum Scale product structure" on page 1
- "Recoverability considerations" on page 40
- "GPFS cluster creation considerations" on page 47
- "IBM Spectrum Scale license designation" on page 80
- "Disk considerations" on page 51
- "File system creation considerations" on page 56

Hardware requirements

You can validate that your hardware meets GPFS requirements by taking the steps outlined in this topic.

1. Consult the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) for the latest list of:
   - Supported server hardware
   - Tested disk configurations
   - Maximum cluster size
2. Provide enough disks to contain the file system. Disks can be:
   - SAN-attached to each node in the cluster
   - Attached to one or more NSD servers
   - A mixture of directly-attached disks and disks that are attached to NSD servers

   Refer to "Network Shared Disk (NSD) creation considerations" on page 52 for additional information.
3. When doing network-based NSD I/O, GPFS passes a large amount of data between its daemons. For NSD server-to-client traffic, it is suggested that you configure a dedicated high-speed network solely for GPFS communications when the following are true:
   - There are NSD disks configured with servers providing remote disk capability.
   - Multiple GPFS clusters are sharing data using NSD network I/O.

   For additional information, see *IBM Spectrum Scale: Advanced Administration Guide*.

GPFS communications require static IP addresses for each GPFS node. IP address takeover operations that transfer the address to another computer are not allowed for the GPFS network. Other IP addresses within the same computer that are not used by GPFS can participate in IP takeover. To provide

© Copyright IBM Corp. 2014, 2016

availability or additional performance, GPFS can use virtual IP addresses created by aggregating several network adapters using techniques such as EtherChannel or channel bonding.

Software requirements

GPFS planning includes understanding the latest software requirements.

- GPFS is supported on AIX, Linux, and Windows.
- For existing GPFS 3.5 clusters, OpenSSL is required for remote cluster access.
- Kernel development files and compiler utilities are required to build the GPFS portability layer on Linux nodes. The required RPMs or packages for each supported Linux distribution are:

  SLES Linux RPMs

  RedHat Linux RPMs
  kernel-default-devel, cpp, gcc, gcc-c++, binutils, ksh

  Note: If required RPMs are not detected, error messages will display the names of the missing RPMs.

  kernel-devel, cpp, gcc, gcc-c++, binutils

  Debian Linux Packages
  linux-headers, cpp, gcc, gcc-c++, binutils
- To use active file management (AFM), the following is required:
  nfs-utils
- To use CNFS, the following are required:
  ethtool
  nfs-utils
  rpcbind
  psmisc
- To use the **mmchconfig numaMemoryInterleave** parameter, the following is required:
  numactl
- To use IBM Spectrum Scale for object storage, the following are required:
  `selinux-policy-base` at 3.13.1-23 or higher
  `selinux-policy-targeted` at 3.12.1-153 or higher

Consult the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) for the latest list of:

- AIX environments
- Linux distributions
- Linux kernel versions
- Windows environments

Recoverability considerations

Sound file system planning requires several decisions about recoverability. After you make these decisions, GPFS parameters enable you to create a highly-available file system with rapid recovery from failures.

- At the disk level, consider preparing disks for use with your file system by specifying failure groups that are associated with each disk. With this configuration, information is not vulnerable to a single point of failure. See “Network Shared Disk (NSD) creation considerations” on page 52.
- At the file system level, consider replication through the metadata and data replication parameters. See “File system replication parameters” on page 63.

Additionally, GPFS provides several layers of protection against failures of various types:

1. "Node failure"
2. "Network Shared Disk server and disk failure" on page 44
3. "Reduced recovery time using Persistent Reserve" on page 46

Node failure

In the event of a node failure, GPFS prevents the continuation of I/O from the failing node and replays the file system metadata log for the failed node.

GPFS prevents the continuation of I/O from a failing node through a GPFS-specific fencing mechanism called *disk leasing*. When a node has access to file systems, it obtains disk leases that allow it to submit I/O. However, when a node fails, that node cannot obtain or renew a disk lease. When GPFS selects another node to perform recovery for the failing node, it first waits until the disk lease for the failing node expires. This allows for the completion of previously submitted I/O and provides for a consistent file system metadata log. Waiting for the disk lease to expire also avoids data corruption in the subsequent recovery step.

To reduce the amount of time it takes for disk leases to expire, you can use Persistent Reserve (SCSI-3 protocol). If Persistent Reserve (configuration parameter: **usePersistentReserve**) is enabled, GPFS prevents the continuation of I/O from a failing node by fencing the failed node using a feature of the disk subsystem called Persistent Reserve. Persistent Reserve allows the failing node to recover faster because GPFS does not need to wait for the disk lease on the failing node to expire. For additional information, refer to "Reduced recovery time using Persistent Reserve" on page 46. For further information about recovery from node failure, see the *IBM Spectrum Scale: Problem Determination Guide*.

File system recovery from node failure should not be noticeable to applications running on other nodes. The only noticeable effect may be a delay in accessing objects that were being modified on the failing node when it failed. Recovery involves rebuilding metadata structures which may have been under modification at the time of the failure. If the failing node is acting as the file system manager when it fails, the delay will be longer and proportional to the level of activity on the file system at the time of failure. In this case, the failover file system management task happens automatically to a surviving node.

Quorum:

GPFS uses a cluster mechanism called quorum to maintain data consistency in the event of a node failure.

Quorum operates on the principle of majority rule. This means that a majority of the nodes in the cluster must be successfully communicating before any node can mount and access a file system. This keeps any nodes that are cut off from the cluster (for example, by a network failure) from writing data to the file system.

During node failure situations, quorum needs to be maintained in order for the cluster to remain online. If quorum is not maintained due to node failure, GPFS unmounts local file systems on the remaining nodes and attempts to reestablish quorum, at which point file system recovery occurs. For this reason it is important that the set of quorum nodes be carefully considered (refer to "Selecting quorum nodes" on page 44 for additional information).

GPFS quorum must be maintained within the cluster for GPFS to remain active. If the quorum semantics are broken, GPFS performs recovery in an attempt to achieve quorum again. GPFS can use one of two methods for determining quorum:

- Node quorum
- Node quorum with tiebreaker disks

Node quorum:

Node quorum is the default quorum algorithm for GPFS.

With node quorum:

- Quorum is defined as one plus half of the *explicitly defined* quorum nodes in the GPFS cluster.
- There are no default quorum nodes; you must specify which nodes have this role.

For example, in Figure 4, there are three quorum nodes. In this configuration, GPFS remains active as long as there are two quorum nodes available.

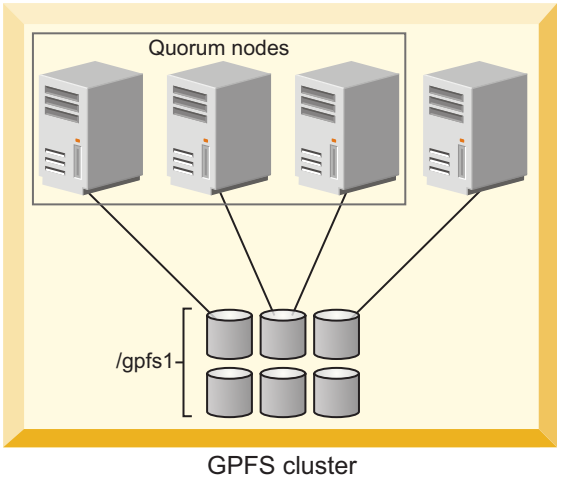


Figure 4. GPFS configuration using node quorum

Node quorum with tiebreaker disks:

When running on small GPFS clusters, you might want to have the cluster remain online with only one surviving node.

To achieve this, you need to add a tiebreaker disk to the quorum configuration. Node quorum with tiebreaker disks allows you to run with as little as one quorum node available as long as you have access to a majority of the quorum disks (refer to Figure 5 on page 44). Enabling node quorum with tiebreaker disks starts by designating one or more nodes as quorum nodes. Then one to three disks are defined as tiebreaker disks using the **tiebreakerDisks** parameter on the **mmchconfig** command. You can designate any disk to be a tiebreaker.

When utilizing node quorum with tiebreaker disks, there are specific rules for cluster nodes and for tiebreaker disks.

Cluster node rules:

1. There is a maximum of eight quorum nodes.
2. All quorum nodes need to have access to all of the tiebreaker disks.
3. When using the traditional server-based (non-CCR) configuration repository, you should include the primary and secondary cluster configuration servers as quorum nodes.
4. You may have an unlimited number of non-quorum nodes.
5. If a network connection fails, which causes the loss of quorum, and quorum is maintained by tiebreaker disks, the following rationale is used to re-establish quorum. If a group has the cluster manager, it is the "survivor". The cluster manager can give up its role if it communicates with fewer

than the minimum number of quorum nodes as defined by the **minQuorumNodes** configuration parameter. In this case, other groups with the minimum number of quorum nodes (if they exist) can choose a new cluster manager.

Changing quorum semantics:

When using the cluster configuration repository (CCR) to store configuration files, the total number of quorum nodes is limited to eight, regardless of quorum semantics, but the use of tiebreaker disks can be enabled or disabled at any time by issuing an **mmchconfig tiebreakerDisks** command. The change will take effect immediately, and it is not necessary to shut down GPFS when making this change.

When using the traditional server-based (non-CCR) configuration repository, it is possible to define more than eight quorum nodes, but only when no tiebreaker disks are defined:

1. To configure more than eight quorum nodes under the server-based (non-CCR) configuration repository, you must disable node quorum with tiebreaker disks and restart the GPFS daemon. To disable node quorum with tiebreaker disks:
   a. Issue the **mmshutdown -a** command to shut down GPFS on all nodes.
   b. Change quorum semantics by issuing **mmchconfig tiebreakerdisks=no**.
   c. Add additional quorum nodes.
   d. Issue the **mmstartup -a** command to restart GPFS on all nodes.
2. If you remove quorum nodes and the new configuration has less than eight quorum nodes, you can change the configuration to node quorum with tiebreaker disks. To enable quorum with tiebreaker disks:
   a. Issue the **mmshutdown -a** command to shut down GPFS on all nodes.
   b. Delete the appropriate quorum nodes or run **mmchnode --nonquorum** to drop them to a client.
   c. Change quorum semantics by issuing the **mmchconfig tiebreakerdisks="***diskList***"** command.
      - The *diskList* contains the names of the tiebreaker disks.
      - The list contains the NSD names of the disks, preferably one or three disks, separated by a semicolon (;) and enclosed by quotes.
   d. Issue the **mmstartup -a** command to restart GPFS on all nodes.

Tiebreaker disk rules:

- You can have one, two, or three tiebreaker disks. However, you should use an odd number of tiebreaker disks.
- Among the quorum node groups that appear after an interconnect failure, only those having access to a majority of tiebreaker disks can be candidates to be the survivor group.
- Tiebreaker disks must be connected to all quorum nodes.

In Figure 5 on page 44 GPFS remains active with the minimum of a single available quorum node and two available tiebreaker disks.

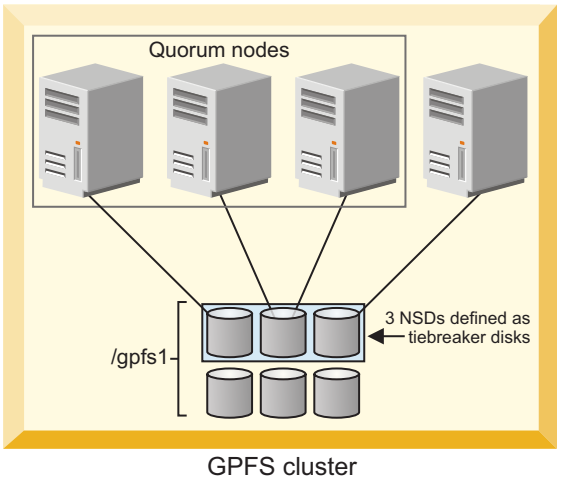


Figure 5. GPFS configuration using node quorum with tiebreaker disks

When a quorum node detects loss of network connectivity, but before GPFS runs the algorithm that decides if the node will remain in the cluster, the **tiebreakerCheck** event is triggered. This event is generated only in configurations that use quorum nodes with tiebreaker disks. It is also triggered on the cluster manager periodically by a challenge-response thread to verify that the node can still continue as cluster manager.

Selecting quorum nodes:

To configure a system with efficient quorum nodes, follow these rules.

- Select nodes that are likely to remain active
  - If a node is likely to be rebooted or require maintenance, do not select that node as a quorum node.
- Select nodes that have different failure points such as:
  - Nodes located in different racks
  - Nodes connected to different power panels
- You should select nodes that GPFS administrative and serving functions rely on such as:
  - Primary configuration servers
  - Secondary configuration servers
  - Network Shared Disk servers
- Select an odd number of nodes as quorum nodes
  - The suggested maximum is seven quorum nodes.
- Having a large number of quorum nodes may increase the time required for startup and failure recovery.
  - Having more than seven quorum nodes does not guarantee higher availability.

Network Shared Disk server and disk failure

The three most common reasons why data becomes unavailable are disk failure, disk server failure with no redundancy, and failure of a path to the disk.

In the event of a disk failure in which GPFS can no longer read or write to the disk, GPFS will discontinue use of the disk until it returns to an available state. You can guard against loss of data availability from disk failure by:

- Utilizing hardware data protection as provided by a Redundant Array of Independent Disks (RAID) device (see Figure 6 on page 45)

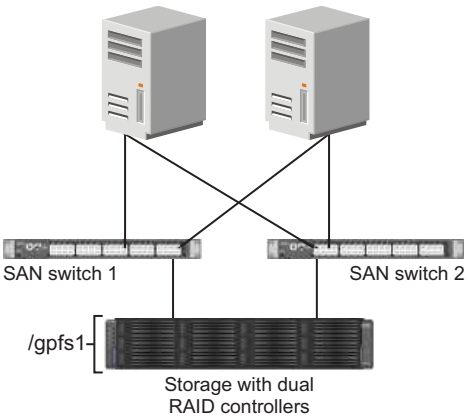


Figure 6. An example of a highly available SAN configuration for a GPFS file system

- Utilizing the GPFS data and metadata replication features (see "Increased data availability" on page 3) along with the designation of failure groups (see "Network Shared Disk (NSD) creation considerations" on page 52 and Figure 7)

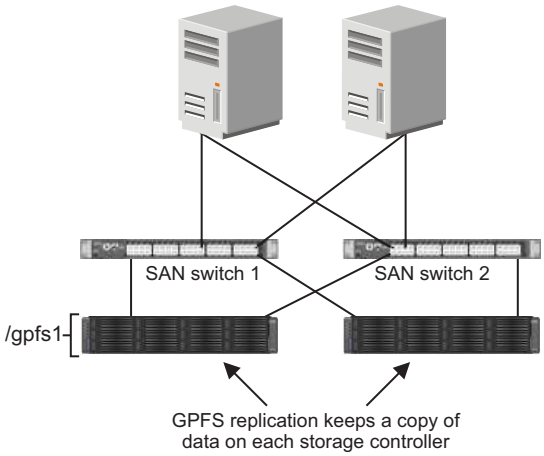


Figure 7. Configuration using GPFS replication for improved availability

It is suggested that you consider RAID as the first level of redundancy for your data and add GPFS replication if you desire additional protection.

In the event of an NSD server failure in which a GPFS client can no longer contact the node that provides remote access to a disk, GPFS discontinues the use of the disk. You can guard against loss of an NSD server availability by using common disk connectivity on multiple NSD server nodes and specifying multiple Network Shared Disk servers for each common disk.

Note: In the event that a path to a disk fails, GPFS reports a disk failure and marks the disk **down**. To bring the disk back online, first follow the directions supplied by your storage vendor to determine and repair the failure.

Guarding against loss of data availability due to path failure

You can guard against loss of data availability from failure of a path to a disk by doing the following:

- Creating multiple NSD servers for each disk

  As GPFS determines the available connections to disks in the file system, it is recommended that you always define more than one NSD server for each disk. GPFS allows you to define up to eight NSD

servers for each NSD. In a SAN configuration where NSD servers have also been defined, if the physical connection is broken, GPFS dynamically switches to the next available NSD server (as defined on the server list) and continues to provide data. When GPFS discovers that the path has been repaired, it moves back to local disk access. This is the default behavior, which can be changed by designating file system mount options. For example, if you never want a node to use the NSD server path to a disk, even if the local path fails, you can set the **-o useNSDserver** mount option to **never**. You can set the mount option using the **mmchfs**, **mmmount**, **mmremotefs**, and **mount** commands.

Important: In Linux on z Systems, it is mandatory to have multiple paths to one SCSI disk (LUN) to avoid single path of failure. The coalescing of the paths to one disk is done by the kernel (via the device-mapper component). As soon as the paths are coalesced, a new logical, multipathed device is created, which is used for any further (administration) task. (The single paths can no longer be used.)

The multipath device interface name depends on the distribution and is configurable:

SUSE **/dev/mapper/***Unique_WW_Identifier*

For example: `/dev/mapper/36005076303ffc56200000000000010cc`

Red Hat
/dev/mapper/mpath*

To obtain information about a multipathed device, use the multipath tool as shown in the following example:

```
# multipath -ll
```

The system displays output similar to this:

```
36005076303ffc56200000000000010cc dm-0 IBM,2107900
[size=5.0G][features=1 queue_if_no_path][hwhandler=0]
\_ round-robin 0 [prio=2][active]
 \_ 1:0:0:0 sdb   8:16  [active][ready]
 \_ 0:0:0:0 sda   8:0   [active][ready]
```

See the question, "What considerations are there when setting up DM-MP multipath service" in the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

- Using an I/O driver that provides multiple paths to the disks for failover purposes

  Failover is a path-management algorithm that improves the reliability and availability of a device because the system automatically detects when one I/O path fails and reroutes I/O through an alternate path.

Reduced recovery time using Persistent Reserve

Persistent Reserve (PR) provides a mechanism for reducing recovery times from node failures.

To enable PR and to obtain recovery performance improvements, your cluster requires a specific environment:

- *All* disks must be PR-capable.
- On AIX, all disks must be hdisks. Starting with 3.5.0.16, it is also possible to use a logical volume as a **descOnly** disk without disabling the use of Persistent Reserve. See the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

  On Linux, typically all disks must be generic (`/dev/sd*`) or DM-MP (`/dev/dm-*`) disks.

  However, for Linux on z Systems, multipath device names are required for SCSI disks, and the names are dependent on the distribution and are configurable. For details, see "Guarding against loss of data availability due to path failure" on page 45.
- If the disks have defined NSD servers, all NSD server nodes must be running the same operating system (AIX or Linux).

- If the disks are SAN-attached to all nodes, all nodes in the cluster must be running the same operating system (AIX or Linux).

You must explicitly enable PR using the **usePersistentReserve** option of the **mmchconfig** command. If you set **usePersistentReserve=yes**, GPFS attempts to set up PR on all of the PR capable disks. All subsequent NSDs are created with PR enabled if they are PR capable. However, PR is only supported in the home cluster. Therefore, access to PR-enabled disks from another cluster must be through an NSD server that is in the home cluster and not directly to the disk (for example, through a SAN).

GPFS cluster creation considerations

A GPFS cluster is created using the **mmcrcluster** command.

Table 4 details the cluster creation options, how to change the options, and the default values for each option.

Table 4. GPFS cluster creation options

Options	Command to change the option	Default value
"Nodes in your GPFS cluster" on page 48	**mmaddnode** **mmdelnode**	None
Node designation: **manager** or **client**, see "Nodes in your GPFS cluster" on page 48	**mmchnode**	**client**
Node designation: **quorum** or **nonquorum**, see "Nodes in your GPFS cluster" on page 48	**mmchnode**	**nonquorum**
Primary cluster configuration server, see "GPFS cluster configuration servers" on page 49	**mmchcluster**	None
Secondary cluster configuration server, see "GPFS cluster configuration servers" on page 49	**mmchcluster**	None
"Remote shell command" on page 49	**mmchcluster**	**/usr/bin/ssh**
"Remote file copy command" on page 50	**mmchcluster**	**/usr/bin/scp**
"Cluster name" on page 50	**mmchcluster**	The node name of the primary GPFS cluster configuration server
GPFS administration adapter port name, see "GPFS node adapter interface names"	**mmchnode**	Same as the GPFS communications adapter port name
GPFS communications adapter port name, see "GPFS node adapter interface names"	**mmchnode**	None
"User ID domain for the cluster" on page 50	**mmchconfig**	The name of the GPFS cluster
"Starting GPFS automatically" on page 50	**mmchconfig**	**no**
"Cluster configuration file" on page 51	Not applicable	None

GPFS node adapter interface names

An adapter interface name refers to the hostname or IP address that GPFS uses to communicate with a node. Specifically, the hostname or IP address identifies the communications adapter over which the GPFS daemons or GPFS administration commands communicate.

The administrator can specify two node adapter interface names for each node in the cluster:

GPFS node name
Specifies the name of the node adapter interface to be used by the GPFS daemons for internode communication.

GPFS admin node name
Specifies the name of the node adapter interface to be used by GPFS administration commands when communicating between nodes. If not specified, the GPFS administration commands use the same node adapter interface used by the GPFS daemons.

These names can be specified by means of the node descriptors passed to the **mmaddnode** or **mmcrcluster** command and can later be changed with the **mmchnode** command.

If multiple adapters are available on a node, this information can be communicated to GPFS by means of the **subnets** parameter on the **mmchconfig** command.

Nodes in your GPFS cluster

When you create your GPFS cluster, you must provide a file containing a list of node descriptors, for each node to be included in the cluster.

The node descriptors can be included in the command line, or they can be contained in a separate node descriptor file with one node definition per line. Each descriptor must be specified in this form:

```
NodeName:NodeDesignations:AdminNodeName
```

NodeName is a required parameter. *NodeDesignations* and *AdminNodeName* are optional parameters.

NodeName
The host name or IP address of the node for GPFS daemon-to-daemon communication.

The host name or IP address that is used for a node must refer to the communication adapter over which the GPFS daemons communicate. Alias names are not allowed. You can specify an IP address at NSD creation, but it will be converted to a host name that must match the GPFS node name. You can specify a node using any of these forms:

- Short hostname (for example, h135n01)
- Long hostname (for example, h135n01.frf.ibm.com)
- IP address (for example, 7.111.12.102)

Note: Host names should always include at least one alpha character, and they should not start with a hyphen (-).

Whichever form you specify, the other two forms must be defined correctly in DNS or the hosts file.

NodeDesignations
An optional, "-" separated list of node roles.

- **manager** | **client** – Indicates whether a node is part of the node pool from which file system managers and token managers can be selected. The default is **client**, which means do not include the node in the pool of manager nodes. For detailed information on the manager node functions, see "The file system manager" on page 10.

  In general, it is a good idea to define more than one node as a manager node. How many nodes you designate as manager depends on the workload and the number of GPFS server licenses you have. If you are running large parallel jobs, you may need more manager nodes than in a four-node cluster supporting a Web application. As a guide, in a large system there should be a different file system manager node for each GPFS file system.
- **quorum** | **nonquorum** – This designation specifies whether or not the node should be included in the pool of nodes from which quorum is derived. The default is **nonquorum**. You need to designate at least one node as a quorum node. It is recommended that you designate at least the primary and secondary cluster configuration servers and NSD servers as quorum nodes.

How many quorum nodes you designate depends upon whether you use node quorum or node quorum with tiebreaker disks. See "Quorum" on page 41.

AdminNodeName
Specifies an optional field that consists of a node name to be used by the administration commands to communicate between nodes.

If *AdminNodeName* is not specified, the *NodeName* value is used.

Follow these rules when adding nodes to your GPFS cluster:

- While a node may mount file systems from multiple clusters, the node itself may only reside in a single cluster. Nodes are added to a cluster using the **mmcrcluster** or **mmaddnode** command.
- The nodes must be available when they are added to a cluster. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and then issue the **mmaddnode** command to add those nodes.
- Designate at least one but not more than seven nodes as quorum nodes. When not using tiebreaker disks, you can designate more quorum nodes, but it is recommended to use fewer than eight if possible. It is recommended that you designate the cluster configuration servers as quorum nodes. How many quorum nodes altogether you will have depends on whether you intend to use the node quorum with tiebreaker algorithm or the regular node based quorum algorithm. For more details, see "Quorum" on page 41.

GPFS cluster configuration servers

You must designate one of the nodes in your GPFS cluster as the primary GPFS cluster configuration server, where GPFS configuration information is maintained. It is strongly suggested that you also specify a secondary GPFS cluster configuration server.

If you do not specify a secondary cluster configuration server:

1. If your primary server fails, the GPFS cluster configuration data files are inaccessible, which results in the failure of any GPFS administration commands that are issued. Similarly, when the GPFS daemon starts up, at least one of the two GPFS cluster configuration servers must be accessible. See "Cluster configuration data files" on page 25.
2. If the primary server fails, you can use the **mmchcluster** command with the **-p** option to designate another node as the primary cluster configuration server. Similarly, you can use the **mmchcluster** command with the **-s** option to define a secondary cluster configuration server. For more information about the **mmchcluster** command, see *IBM Spectrum Scale: Administration and Programming Reference*.

Remote shell command

GPFS commands need to be able to communicate across all nodes in the cluster. To achieve this, the GPFS commands use the remote shell command that you specify on the **mmcrcluster** command or the **mmchcluster** command.

The default remote shell command is **ssh**. You can designate the use of a different remote shell command by specifying its fully-qualified path name on the **mmcrcluster** command or the **mmchcluster** command. The remote shell command must adhere to the same syntax as **ssh**, but it can implement an alternate authentication mechanism.

Clusters that include both UNIX and Windows nodes must use **ssh** for the remote shell command. For more information, see "Installing and configuring OpenSSH" on page 168.

Clusters that only include Windows nodes may use the **mmwinrsh** utility that comes with GPFS. The fully-qualified path name is **/usr/lpp/mmfs/bin/mmwinrsh**. For more information about configuring Windows GPFS clusters, see **mmwinservctl command** in *IBM Spectrum Scale: Administration and Programming Reference*.

By default, you can issue GPFS administration commands from any node in the cluster. Optionally, you can choose a subset of the nodes that are capable of running administrative commands. In either case, the nodes that you plan to use for administering GPFS must be able to run remote shell commands on any other node in the cluster as user **root** without the use of a password and without producing any extraneous messages.

For additional information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration and Programming Reference*.

Remote file copy command

The GPFS commands must maintain a number of configuration files across all nodes in the cluster. To achieve this, the GPFS commands use the remote file copy command that you specify on the **mmcrcluster** command or the **mmchcluster** command.

The default remote file copy program is **rcp**. You can designate the use of a different remote file copy command by specifying its fully-qualified path name on the **mmcrcluster** command or the **mmchcluster** command. The remote file copy command must adhere to the same syntax as **rcp**, but it can implement an alternate authentication mechanism. Many clusters use **scp** instead of **rcp**, as **rcp** cannot be used in a cluster that contains Windows Server nodes.

Clusters that include both UNIX and Windows nodes must use **scp** for the remote copy command. For more information, see "Installing and configuring OpenSSH" on page 168.

Clusters that only include Windows nodes may use the **mmwinrcp** utility that comes with GPFS. The fully-qualified path name is **/usr/lpp/mmfs/bin/mmwinrcp**. For more information about configuring Windows GPFS clusters, see **mmwinservctl command** in *IBM Spectrum Scale: Administration and Programming Reference*.

The nodes that you plan to use for administering GPFS must be able to copy files using the remote file copy command to and from any other node in the cluster without the use of a password and without producing any extraneous messages.

For additional information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration and Programming Reference*.

Cluster name

Provide a name for the cluster by issuing the **-C** option on the **mmcrcluster** command.

If the user-provided name contains dots, it is assumed to be a fully qualified domain name. Otherwise, to make the cluster name unique in a multiple cluster environment, GPFS appends the domain name of the primary cluster configuration server. If the **-C** option is not specified, the cluster name defaults to the hostname of the primary cluster configuration server. The name of the cluster may be changed at a later time by issuing the **-C** option on the **mmchcluster** command.

The cluster name is applicable when GPFS file systems are mounted by nodes belonging to other GPFS clusters. See the **mmauth** and the **mmremotecluster** commands.

User ID domain for the cluster

This option is the user ID domain for a cluster when accessing a file system remotely.

This option is further explained in *IBM Spectrum Scale: Advanced Administration Guide* and 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).

Starting GPFS automatically

You can specify whether to start the GPFS daemon automatically on a node when it is started.

Whether or not GPFS automatically starts is determined using the **autoload** parameter of the **mmchconfig** command. The default is *not* to automatically start GPFS on all nodes. You may change this by specifying **autoload=yes** using the **mmchconfig** command. This eliminates the need to start GPFS by issuing the **mmstartup** command when a node is booted.

The autoload parameter can be set the same or differently for each node in the cluster. For example, it may be useful to set **autoload=no** on a node that is undergoing maintenance since operating system upgrades and other software can often require multiple reboots to be completed.

Cluster configuration file

GPFS provides default configuration values, so a cluster configuration file is not required to create a cluster.

This optional file can be useful if you already know the correct parameter values based on previous testing or if you are restoring a cluster and have a backup copy of configuration values that apply to most systems. Typically, however, this option is not used at cluster creation time, and configuration parameters are modified after the cluster is created (using the **mmchconfig** command).

Disk considerations

Designing a proper storage infrastructure for your GPFS file systems is key to achieving performance and reliability goals. When deciding what disk configuration to use, you should consider three key areas: infrastructure, performance, and disk access method.

Infrastructure

- Ensure that you have sufficient disks to meet the expected I/O load. In GPFS terminology, a disk may be a physical disk or a RAID device.
- Ensure that you have sufficient connectivity (adapters and buses) between disks and network shared disk servers.
- Determine whether you are within GPFS limits. Starting with GPFS 3.1, the structural limit on the maximum number of disks in a file system increased from 2048 to 4096; however, the current version of GPFS still enforces the original limit of 2048. Should your environment require support for more than 2048 disks, contact the IBM Support Center to discuss increasing the enforced limit. (However, the number of disks in your system is often constrained by products other than GPFS.)
- For a list of storage devices tested with GPFS, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
- For Linux on z Systems: See the “Storage” topics “DASD device driver” and “SCSI-over-Fibre Channel device driver” in Device Drivers, Features, and Commands (www.ibm.com/support/knowledgecenter/api/content/linuxonibm/liaaf/lnz_r_dd.html) in the Linux on z Systems library overview.

Disk access method

- Decide how your disks will be connected. Supported types of disk connectivity include the following configurations:
  1. All disks SAN-attached to all nodes in all clusters that access the file system

     In this configuration, every node sees the same disk simultaneously and has a corresponding disk device entry.
  2. Each disk connected to multiple NSD server nodes (up to eight servers), as specified on the server list

     In this configuration, a single node with connectivity to a disk performs data shipping to all other nodes. This node is the first NSD server specified on the NSD server list. You can define additional NSD servers on the server list. Having multiple NSD servers guards against the loss of a single NSD server. When using multiple NSD servers, all NSD servers must have connectivity to the same disks. In this configuration, all nodes that are not NSD

servers will receive their data over the local area network from the first NSD server on the server list. If the first NSD server fails, the next available NSD server on the list will control data distribution.

3. A combination of SAN-attached and an NSD server configuration.

   Configuration consideration:

   - If the node has a physical attachment to the disk and that connection fails, the node switches to using a specified NSD server to perform I/O. For this reason, it is recommended that you define NSDs with multiple servers, even if all nodes have physical attachments to the disk.
   - Configuring GPFS disks without an NSD server stops the serving of data when the direct path to the disk is lost. This may be a preferable option for nodes requiring a higher speed data connection provided through a SAN as opposed to a lower speed network NSD server connection. Parallel jobs using MPI often have this characteristic.
   - The **-o useNSDserver** file system mount option on the **mmmount**, **mount**, **mmchfs**, and **mmremotefs** commands can be used to specify the disk discovery, and limit or eliminate switching from local access to NSD server access, or the other way around.

- Decide whether you will use storage pools to manage your disks.

  Storage pools allow you to manage your file system's storage in groups. You can partition your storage based on such factors as performance, locality, and reliability. Files are assigned to a storage pool based on defined policies.

  Policies provide for the following:

  - Placing files in a specific storage pool when the files are created
  - Migrating files from one storage pool to another
  - File deletion based on file characteristics

  See the *IBM Spectrum Scale: Advanced Administration Guide* for more information.

Disk considerations include:

1. "Network Shared Disk (NSD) creation considerations"
2. "NSD server considerations" on page 53
3. "File system descriptor quorum" on page 54
4. "Preparing direct access storage devices (DASD) for NSDs" on page 55

Network Shared Disk (NSD) creation considerations

You must prepare each physical disk you intend to use with GPFS by first defining it as a Network Shared Disk (NSD) using the **mmcrnsd** command.

On Windows, GPFS will only create NSDs from empty disk drives. **mmcrnsd** accepts Windows *Basic* disk or *Unknown/Not Initialized* disks. It always re-initializes these disks so that they become *Basic GPT Disks* with a single *GPFS partition*. NSD data is stored in GPFS partitions. This allows other operating system components to recognize that the disks are used. **mmdelnsd** deletes the partition tables created by **mmcrnsd**.

A new NSD format was introduced with GPFS 4.1. The new format is referred to as NSD v2, and the old format is referred to as NSD v1. The NSD v1 format is compatible with GPFS releases prior to 4.1. The latest GPFS release recognizes both NSD v1 and NSD v2 formatted disks.

The NSD v2 format provides the following benefits:

- On Linux, includes a partition table so that the disk is easily recognized as a GPFS device
- Adjusts data alignment to support disks with a 4 KB physical block size

- Adds backup copies of some key GPFS data structures
- Expands some reserved areas to allow for future growth

Administrators do not need to select one format or the other when managing NSDs. GPFS will always create and use the correct format based on the **minReleaseLevel** for the cluster and the file system version. When **minReleaseLevel** (as reported by **mmlsconfig**) is less than 4.1.0.0, **mmcrnsd** will only create NSD v1 formatted disks. When **minReleaseLevel** is at least 4.1.0.0, **mmcrnsd** will only create NSD v2 formatted disks. In this second case, however, the NSD format may change dynamically when the NSD is added to a file system so that the NSD is compatible with the file system version.

On Linux, NSD v2 formatted disks include a GUID Partition Table (GPT) with a single partition. The GPT allows other operating system utilities to recognize when a disk is owned by GPFS, which helps prevent inadvertent data corruption. After running **mmcrnsd**, Linux utilities like **parted** can show the partition table. When an NSD v2 formatted disk is added to a 3.5 or older file system, its format is changed to NSD v1 and the partition table is converted to an MBR (MS-DOS compatible) type.

Note: Leftover persistent reserve (PR) keys can cause problems such as reservation conflicts in multipath, which can in turn cause I/O failure. In such cases, it is necessary to clean up leftover PR keys on a fresh install. For a detailed procedure, see *Clearing a leftover Persistent Reserve reservation* in *IBM Spectrum Scale: Problem Determination Guide*.

The **mmcrnsd** command expects as input a stanza file. For details, see the following topics in *IBM Spectrum Scale: Administration and Programming Reference*:

- *Stanza files*
- **mmchdisk** *command*
- **mmchnsd** *command*
- **mmcrfs** *command*
- **mmcrnsd** *command*

NSD server considerations

If you plan to use NSD servers to remotely serve disk data to other nodes, as opposed to having disks SAN-attached to all nodes, you should consider the total computing and I/O load on these nodes.

- Will your Network Shared Disk servers be dedicated servers or will you also be using them to run applications? If you will have non-dedicated servers, consider running less time-critical applications on these nodes. If you run time-critical applications on a Network Shared Disk server, servicing disk requests from other nodes might conflict with the demands of these applications.
- The special functions of the file system manager consume extra processing time. If possible, avoid using a Network Shared Disk server as the file system manager. The Network Shared Disk server consumes both memory and processor cycles that could impact the operation of the file system manager. See "The file system manager" on page 10.
- The actual processing capability required for Network Shared Disk service is a function of the application I/O access patterns, the type of node, the type of disk, and the disk connection. You can later run **iostat** on the server to determine how much of a load your access pattern will place on a Network Shared Disk server.
- Providing sufficient disks and adapters on the system to yield the required I/O bandwidth. Dedicated Network Shared Disk servers should have sufficient disks and adapters to drive the I/O load you expect them to handle.
- Knowing approximately how much storage capacity you will need for your data.

You should consider what you want as the default behavior for switching between local access and NSD server access in the event of a failure. To set this configuration, use the **-o useNSDserver** file system mount option of the **mmmount**, **mount**, **mmchfs**, and **mmremotefs** commands to:

- Specify the disk discovery behavior

- Limit or eliminate switching from either:
  - Local access to NSD server access
  - NSD server access to local access

You should consider specifying how long to wait for an NSD server to come online before allowing a file system mount to fail because the server is not available. The **mmchconfig** command has these options:

nsdServerWaitTimeForMount
When a node is trying to mount a file system whose disks depend on NSD servers, this option specifies the number of seconds to wait for those servers to come up. If a server recovery is taking place, the wait time you are specifying with this option starts after recovery completes.

Note: The decision to wait for servers is controlled by the **nsdServerWaitTimeWindowOnMount** option.

nsdServerWaitTimeWindowOnMount
Specifies a window of time (in seconds) during which a mount can wait for NSD servers as described for the **nsdServerWaitTimeForMount** option. The window begins when quorum is established (at cluster startup or subsequently), or at the last known failure times of the NSD servers required to perform the mount.

Notes:

1. When a node rejoins a cluster, it resets all the failure times it knew about within that cluster.
2. Because a node that rejoins a cluster resets its failure times within that cluster, the NSD server failure times are also reset.
3. When a node attempts to mount a file system, GPFS checks the cluster formation criteria first. If that check falls outside the window, it will then check for NSD server fail times being in the window.

File system descriptor quorum

A GPFS structure called the *file system descriptor* is initially written to every disk in the file system and is replicated on a subset of the disks as changes to the file system occur, such as the adding or deleting of disks.

Based on the number of failure groups and disks, GPFS creates one to five replicas of the descriptor:

- If there are at least five different failure groups, five replicas are created.
- If there are at least three different disks, three replicas are created.
- If there are only one or two disks, a replica is created on each disk.

Once it decides how many replicas to create, GPFS picks disks to hold the replicas, so that all replicas are in different failure groups, if possible, to reduce the risk of multiple failures. In picking replica locations, the current state of the disks is taken into account. Stopped or suspended disks are avoided. Similarly, when a failed disk is brought back online, GPFS might rebalance the file system descriptors in order to assure reliability across the failure groups. The disks used to hold the file system descriptor replicas can be seen by running the **mmlsdisk** *fsname* **-L** command and looking for the string **desc** in the **remarks** column.

GPFS requires that a majority of the replicas on the subset of disks remain available to sustain file system operations:

- If there are at least five different replicas, GPFS can tolerate a loss of two of the five replicas.
- If there are at least three replicas, GPFS can tolerate a loss of one of the three replicas.
- If there are fewer than three replicas, a loss of one replica might make the descriptor inaccessible.

The loss of all disks in a disk failure group might cause a majority of file systems descriptors to become unavailable and inhibit further file system operations. For example, if your file system is backed up by three or more disks that are assigned to two separate disk failure groups, one of the failure groups will be assigned two of the file system descriptor replicas, while the other failure group will be assigned only one replica. If all of the disks in the disk failure group that contains the two replicas were to become unavailable, the file system would also become unavailable. To avoid this particular scenario, you might want to introduce a third disk failure group consisting of a single disk that is designated as a **descOnly** disk. This disk would exist solely to contain a replica of the file system descriptor (that is, it would not contain any file system metadata or data). This disk should be at least 128MB in size.

For more information, see "Network Shared Disk (NSD) creation considerations" on page 52 and *Establishing disaster recovery for your GPFS cluster* in *IBM Spectrum Scale: Advanced Administration Guide*.

Preparing direct access storage devices (DASD) for NSDs

When preparing direct access storage devices (DASD) for NSDs, see the table "Disk hardware tested with GPFS for Linux on z Systems" in the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Preparing your environment for use of extended count key data (ECKD) devices:

If your GPFS cluster includes Linux on z Systems instances, do not use virtual reserve/release.

Instead, follow the process described in Sharing DASD without Using Virtual Reserve/Release (www.ibm.com/support/knowledgecenter/SSB27U_6.3.0/com.ibm.zvm.v630.hcpa5/hcpa5259.htm). Data integrity is handled by GPFS itself.

Preparing an ECKD™ device for GPFS

To prepare an ECKD device for GPFS, complete these steps on a single node:

1. Ensure that the ECKD device is online. To set it online, issue the following command:

   ```
   chccwdev -e device_bus_id
   ```

   where *device_bus_id* identifies the device to be configured. *device_bus_id* is a device number with a leading **0.***n*, where *n* is the subchannel set ID. For example:

   ```
   chccwdev -e 0.0.3352
   ```

2. Low-level format the ECKD using one of the following commands.

   Note: GPFS supports ECKD disks in either compatible disk layout (CDL) format or Linux disk layout (LDL) format. The DASD must be formatted with a block size of 4096.

   - To specify CDL format, issue the following command:

     ```
     dasdfmt -d cdl device
     ```

     There is no need to specify a block size value, as the default value is 4096.
   - To specify LDL format, issue the following command:

     ```
     dasdfmt -d ldl device
     ```

     There is no need to specify a block size value, as the default value is 4096.

   In both of these commands, *device* is the node of the device. For example:

   ```
   dasdfmt -d cdl /dev/dasda
   ```

3. This step is for *CDL disks only*. It is an *optional* step because partitioning is optional for CDL disks.

   If you wish to partition the ECKD and create a single partition that spans the entire device, use the following command:

   ```
   fdasd -a device
   ```

   Notes:

- This step is not required for LDL disks because the **dasdfmt -d ldl** command issued in the previous step automatically creates a single Linux partition on the disk.
- If a CDL disk is partitioned, the partition name should be specified in the stanza input file for **mmcrnsd**. If a CDL disk is not partitioned, the disk name should be specified in the stanza input file.

For more information about all of these commands, see the following:

- "Commands for Linux on z Systems" topic in Device Drivers, Features, and Commands (www.ibm.com/support/knowledgecenter/api/content/linuxonibm/liaaf/lnz_r_dd.html) in the Linux on z Systems library overview.
- "Getting started with Elastic Storage for Linux on z Systems based on GPFS technology" white paper, available on the Welcome Page for IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html).

Repeat these steps for each ECKD to be used with GPFS.

After preparing the environment, set the ECKD devices online on the other nodes.

Note: In the case of an ECKD device that is attached to two servers and is online on both, if you format or partition the disk on one of the servers, the other server does not automatically know about that. You must refresh the information about the disk on the other server. One way to do this is to set the disk offline and then online again.

Always ensure that the ECKD devices are online before starting GPFS. To automatically set ECKD devices online at system start, see the documentation for your Linux distribution.

File system creation considerations

File system creation involves anticipating usage within the file system and considering your hardware configurations. Before creating a file system, consider how much data will be stored and how great the demand for the files in the system will be.

Each of these factors can help you to determine how much disk resource to devote to the file system, which block size to choose, where to store data and metadata, and how many replicas to maintain. For the latest supported file system size, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Your GPFS file system is created by issuing the **mmcrfs** command. Table 5 details the file system creation options specified on the **mmcrfs** command, which options can be changed later with the **mmchfs** command, and what the default values are.

To move an existing file system into a new GPFS cluster, see *Exporting file system definitions between clusters* in *IBM Spectrum Scale: Advanced Administration Guide*.

Table 5. File system creation options

Options	mmcrfs	mmchfs	Default value
Device name of the file system. See "Device name of the file system" on page 59.	X	X	none

Table 5. File system creation options (continued)

Options	mmcrfs	mmchfs	Default value
DiskDesc for each disk in your file system. **Note:** The use of disk descriptors is discouraged. See "Disks for your file system" on page 60.	X	Issue the **mmadddisk** or **mmdeldisk** command to add or delete disks from the file system.	none
-F *StanzaFile* specifies a file that contains a list of NSD stanzas. For more information, see *Stanza files* in *IBM Spectrum Scale: Administration and Programming Reference*.	X	Issue the **mmadddisk** or **mmdeldisk** command to add or delete disks as indicated in the stanza file.	none
-A {**yes** | **no** | **automount**} to determine when to mount the file system. See "Deciding how the file system is mounted" on page 60.	X	X	**yes**
-B *BlockSize* to set the data block size: **64K**, **128K**, **256K**, **512K**, **1M**, **2M**, **4M**, **8M** or **16M**. See "Block size" on page 60.	X	This value cannot be changed without re-creating the file system.	**256K**
-D {**posix** | **nfs4**} semantics for a deny-write open lock See "NFS V4 deny-write open lock" on page 60.	X	X	**nfs4**
-E {**yes** | **no**} to report exact mtime values. See "mtime values" on page 61.	X	X	**yes**
-i *InodeSize* to set the size of inodes: **512**, **1024**, or **4096** bytes.	X	This value cannot be changed.	**4096**
-j {**cluster** | **scatter**} to determine the block allocation map type. See "Block allocation map" on page 62.	X	NA	See "Block allocation map" on page 62.
-k {**posix** | **nfs4** | **all**} to determine the authorization types supported by the file system. See "File system authorization" on page 62.	X	X	**all**
-K {**no** | **whenpossible** | **always**} to enforce strict replication. See "Strict replication" on page 63.	X	X	**whenpossible**
-L *LogFileSize* to specify the size of the internal log files. See "GPFS recovery logs" on page 14.	X	X	The default is **4 MB** or the metadata block size, whichever is larger.

Table 5. File system creation options (continued)

Options	mmcrfs	mmchfs	Default value
-m *DefaultMetadataReplicas* See "File system replication parameters" on page 63.	X	X	**1**
-M *MaxMetadataReplicas* See "File system replication parameters" on page 63.	X	This value cannot be changed.	**2**
-n *NumNodes* that will mount the file system. See "Number of nodes mounting the file system" on page 64.	X	X	**32**
-o *MountOptions* to be passed to the **mount** command. See "Assign mount command options" on page 65.	NA	X	none
-Q {**yes** | **no**} to activate quota. See "Enabling quotas" on page 65.	X	X	**no**
-r *DefaultDataReplicas* See "File system replication parameters" on page 63.	X	X	**1**
-R *MaxDataReplicas* See "File system replication parameters" on page 63.	X	This value cannot be changed.	**2**
-S {**yes** | **no** | **relatime**} to control how the atime value is updated. See "atime values" on page 61.	X	X	**no**
-t *DriveLetter* See "Windows drive letter" on page 64.	X	X	none
-T *Mountpoint* See "Mountpoint directory" on page 64.	X	X	**/gpfs/***DeviceName*
-V {**full** | **compat**} to change the file system format to the latest level. See "Changing the file system format to the latest level" on page 67	NA	X	none
-v {**yes** | **no**} to verify disk usage. See "Verifying disk usage" on page 66.	X	NA	**yes**
-W *NewDeviceName* to assign a new device name to the file system.	NA	X	none
-z {**yes** | **no**} to enable DMAPI See "Enable DMAPI" on page 66.	X	X	**no**

Table 5. File system creation options (continued)

Options	mmcrfs	mmchfs	Default value
--filesetdf to specify (when quotas are enforced for a fileset) whether the **df** command will report numbers based on the quotas for the fileset and not for the total file system.	X	X	**--nofilesetdf**
--inode-limit *MaxNumInodes* [:*NumInodesToPreallocate*] to determine the maximum number of files in the file system. See "Specifying the maximum number of files that can be created" on page 67.	X	X	file system size/**1** MB
--log-replicas *LogReplicas* to specify the number of recovery log replicas.	X	X	none
--metadata-block-size *MetadataBlockSize* to specify the block size for the system storage pool. See "Block size" on page 60.	X	NA	The default is the same as the value set for **-B** *BlockSize*.
--mount-priority *Priority* to control the order in which the individual file systems are mounted at daemon startup or when one of the **all** keywords is specified on the **mmmount** command.	X	X	**0**
--perfileset-quota to set the scope of user and group quota limit checks to the individual fileset level.	X	X	**--noperfileset-quota**
--rapid-repair to keep track of incomplete replication on an individual file block basis (as opposed to the entire file).	NA	X	none
--version *VersionString* to enable only the file system features that are compatible with the specified release. See "Enabling file system features" on page 67.	X	NA	**4.1.1.0**
Notes: 1. X – indicates that the option is available on the command. 2. NA (not applicable) – indicates that the option is not available on the command.			

Device name of the file system

File system names must be unique within a GPFS cluster. However, two different clusters can have two distinct file systems with the same name.

The device name of the file system does not need to be fully qualified. **fs0** is as acceptable as **/dev/fs0**. The name cannot be the same as an existing entry in **/dev**.

Note: If your cluster includes Windows nodes, the file system name should be no longer than 31 characters.

NFS V4 deny-write open lock

You can specify whether a deny-write open lock blocks writes, which is expected and required by NFS V4, Samba, and Windows.

For more information, see *Managing GPFS access control lists* in *IBM Spectrum Scale: Administration and Programming Reference*.

nfs4 Must be specified for file systems supporting NFS V4 and file systems mounted on Windows. This is the default.

posix Specified for file systems supporting NFS V3 or ones which are not NFS exported.

posix allows NFS writes even in the presence of a deny-write open lock.

Disks for your file system

Disks must be defined as NSDs before they can be added to a GPFS file system.

NSDs are created using the **mmcrnsd** command. You can use the **mmlsnsd -F** command to display a list of available NSDs.

See "Disk considerations" on page 51.

Deciding how the file system is mounted

Specify when the file system is to be mounted:

yes When the GPFS daemon starts. This is the default.

no Manual mount.

automount
When the file system is first accessed.

This can be changed at a later time by using the **-A** option on the **mmchfs** command.

Considerations:

1. GPFS mount traffic may be lessened by using the automount feature to mount the file system when it is first accessed instead of at GPFS startup. Automatic mounts only produce additional control traffic at the point that the file system is first used by an application or user. Mounts at GPFS startup on the other hand produce additional control traffic at every GPFS startup. Thus startup of hundreds of nodes at once may be better served by using automatic mounts.
2. Automatic mounts will fail if the node does not have the operating systems automount support enabled for the file system.
3. When exporting file systems for NFS mounts, it may be useful to mount the file system when GPFS starts.

Block size

The size of data blocks in a file system can be specified at file system creation by using the **-B** option on the **mmcrfs** command or allowed to default to 256 KB. This value *cannot* be changed without re-creating the file system.

GPFS supports these block sizes for file systems: 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, 8 MB and 16 MB (for GPFS Native RAID only). This value should be specified with the character **K** or **M** as appropriate, for example: `512K` or `4M`. You should choose the block size based on the application set that you plan to support and whether you are using RAID hardware. For more information about GPFS Native RAID, see *IBM Spectrum Scale RAID: Administration* in the Elastic Storage Server documentation on IBM Knowledge Center.

The **--metadata-block-size** option on the **mmcrfs** command allows a different block size to be specified for the system storage pool, provided its usage is set to **metadataOnly**. This can be especially beneficial if the default block size is larger than 1 MB. Valid values are the same as those listed for the **-B** option.

If you plan to use RAID devices in your file system, a larger block size may be more effective and help avoid the penalties involved in small block write operations to RAID devices. For example, in a RAID configuration using 4 data disks and 1 parity disk (a 4+P RAID 5 configuration), which uses a 64 KB stripe size, the optimal file system block size would be an integral multiple of 256 KB (4 data disks × 64 KB stripe size = 256 KB). A block size of an integral multiple of 256 KB results in a single data **write** that encompasses the 4 data disks and a parity-write to the parity disk. If a block size smaller than 256 KB, such as 64 KB, is used with the same RAID configuration, **write** performance is degraded by the read-modify-write behavior. A 64 KB block size results in a single disk writing 64 KB and a subsequent **read** from the three remaining disks in order to compute the parity that is then written to the parity disk. The extra **read** degrades performance.

The choice of block size also affects the performance of certain metadata operations, in particular, block allocation performance. The GPFS block allocation map is stored in blocks, similar to regular files. When the block size is small:

- It takes more blocks to store a given amount of data resulting in additional work to allocate those blocks
- One block of allocation map data contains less information

Note: The choice of block size is particularly important for large file systems. For file systems larger than 100 TB, you should use a block size of at least 256 KB.

Fragments and subblocks:

GPFS divides each block into 32 *subblocks*. Files smaller than one block size are stored in *fragments*, which are made up of one or more subblocks. Large files are stored in a number of full blocks plus zero or more subblocks to hold the data at the end of the file.

The block size is the largest contiguous amount of disk space allocated to a file and therefore the largest amount of data that can be accessed in a single I/O operation. The subblock is the smallest unit of disk space that can be allocated. For a block size of 256 KB, GPFS reads as much as 256 KB of data in a single I/O operation and small files can occupy as little as 8 KB of disk space.

atime values

atime is a standard file attribute that represents the time when the file was last accessed.

The **-S** file system configuration parameter controls how the **atime** value is updated. The default is **-S no**, which results in updating **atime** locally in memory whenever a file is read, but the value is not visible to other nodes until after the file is closed. If an accurate **atime** value is needed, the application must use the GPFS calls **gpfs_stat()** and **gpfs_fstat()** functions. When **-S yes** is specified, or the file system is mounted **read-only**, the updating of the **atime** value is suppressed. This means that the **atime** value is no longer updated. This can be an issue if you have policies that use the **ACCESS_TIME** file attribute for file management. For more information, see *Exceptions to the Open Group technical standards* in *IBM Spectrum Scale: Administration and Programming Reference*.

When **-S relatime** is specified, the file access time is updated only if the existing access time is older than the value of the **atimeDeferredSeconds** configuration attribute or the existing file modification time is greater than the existing access time.

mtime values

mtime is a standard file attribute that represents the time when the file was last modified.

The **-E** parameter controls when the **mtime** is updated. The default is **-E yes**, which results in standard interfaces including the **stat()** and **fstat()** calls reporting exact **mtime** values. Specifying **-E no** results in the **stat()** and **fstat()** calls reporting the **mtime** value available at the completion of the last sync period. This may result in the calls not always reporting the exact **mtime**. Setting **-E no** can affect backup operations that rely on last modified time or the operation of policies using the **MODIFICATION_TIME** file attribute.

For more information, see *Exceptions to the Open Group technical standards* in *IBM Spectrum Scale: Administration and Programming Reference*.

Block allocation map

GPFS has two different methods of allocating space in a file system. The **-j** parameter specifies the block allocation map type to use when creating a file system. The block allocation map type cannot be changed once the file system is created.

When allocating blocks for a given file, GPFS first uses a round-robin algorithm to spread the data across all of the disks in the file system. After a disk is selected, the location of the data block on the disk is determined by the block allocation map type.

The two types of allocation methods are **cluster** and **scatter**:

cluster
: GPFS attempts to allocate blocks in clusters. Blocks that belong to a given file are kept next to each other within each cluster.

  This allocation method provides better disk performance for some disk subsystems in relatively small installations. The benefits of clustered block allocation diminish when the number of nodes in the cluster or the number of disks in a file system increases, or when the file system free space becomes fragmented. The **cluster** allocation method is the default for GPFS clusters with eight or fewer nodes and for file systems with eight or fewer disks.

scatter
: GPFS chooses the location of the blocks randomly.

  This allocation method provides more consistent file system performance by averaging out performance variations due to block location (for many disk subsystems, the location of the data relative to the disk edge has a substantial effect on performance). This allocation method is appropriate in most cases and is the default for GPFS clusters with more than eight nodes or file systems with more than eight disks.

This parameter for a given file system is specified at file system creation by using the **-j** option on the **mmcrfs** command, or allowing it to default. This value *cannot* be changed after the file system has been created.

File system authorization

The type of authorization for the file system is specified on the **-k** option on the **mmcrfs** command or changed at a later time by using the **-k** option on the **mmchfs** command.

posix
: Traditional GPFS access control lists (ACLs) only (NFS V4 and Windows ACLs are not allowed).

nfs4
: Support for NFS V4 and Windows ACLs only. Users are not allowed to assign traditional ACLs to any file system objects.

all
: Allows for the coexistence of POSIX, NFS V4, and Windows ACLs within a file system. This is the default.

If the file system will be used for protocol exports (NFS/SMB/Swift), **nfs4** is recommended.

Strict replication

Strict replication means that data or metadata replication is performed at all times, according to the replication parameters specified for the file system. If GPFS cannot perform the file system's replication, an error is returned.

These are the choices:

no Strict replication is not enforced. GPFS tries to create the needed number of replicas, but returns an **errno** of EOK if it can allocate at least one replica.

whenpossible
Strict replication is enforced if the disk configuration allows it. If the number of failure groups is insufficient, strict replication is not enforced. This is the default value.

always
Indicates that strict replication is enforced.

The use of strict replication can be specified at file system creation by using the **-K** option on the **mmcrfs** command. The default is **whenpossible**. This value can be changed using the **mmchfs** command.

Internal log file

You can specify the internal log file size. Refer to "GPFS recovery logs" on page 14 for additional information.

File system replication parameters

The metadata (inodes, directories, and indirect blocks) and data replication parameters are set at the file system level and apply to all files. They are initially set for the file system when issuing the **mmcrfs** command.

They can be changed for an existing file system using the **mmchfs** command. When the replication parameters are changed, files created after the change are affected. To apply the new replication values to existing files in a file system, issue the **mmrestripefs** command.

Metadata and data replication are specified independently. Each has a default replication factor of 1 (no replication) and a maximum replication factor with a default of 2. Although replication of metadata is less costly in terms of disk space than replication of file data, excessive replication of metadata also affects GPFS efficiency because all metadata replicas must be written. In general, more replication uses more space.

Default metadata replicas:

The default number of copies of metadata for all files in the file system may be specified at file system creation by using the **-m** option on the **mmcrfs** command or changed at a later time by using the **-m** option on the **mmchfs** command.

This value must be equal to or less than *MaxMetadataReplicas*, and cannot exceed the number of failure groups with disks that can store metadata. The allowable values are 1, 2, or 3, with a default of 1 (no replication).

Maximum metadata replicas:

The maximum number of copies of metadata for all files in the file system can be specified at file system creation by using the **-M** option on the **mmcrfs** command.

The default is 2. The allowable values are 1, 2, or 3, but it cannot be less than the value of *DefaultMetadataReplicas*. This value cannot be changed after the file system is created.

Default data replicas:

The default replication factor for data blocks may be specified at file system creation by using the **-r** option on the **mmcrfs** command or changed at a later time by using the **-r** option on the **mmchfs** command.

This value must be equal to or less than *MaxDataReplicas*, and the value cannot exceed the number of failure groups with disks that can store data. The allowable values are 1, 2, and 3, with a default of 1 (no replication).

If you want to change the data replication factor for the entire file system, the data disk in each storage pool must have a number of failure groups equal to or greater than the replication factor. For example, you will get a failure with error messages if you try to change the replication factor for a file system to 2 but the storage pool has only one failure group.

Maximum data replicas:

The maximum number of copies of data blocks for a file can be specified at file system creation by using the **-R** option on the **mmcrfs** command. The default is 2.

The allowable values are 1, 2, and 3, but cannot be less than the value of *DefaultDataReplicas*. This value cannot be changed.

Number of nodes mounting the file system

The estimated number of nodes that will mount the file system may be specified at file system creation by using the **-n** option on the **mmcrfs** command or allowed to default to 32.

When creating a GPFS file system, over-estimate the number of nodes that will mount the file system. This input is used in the creation of GPFS data structures that are essential for achieving the maximum degree of parallelism in file system operations (see "GPFS architecture" on page 9). Although a larger estimate consumes a bit more memory, insufficient allocation of these data structures can limit the ability to process certain parallel requests efficiently, such as the allotment of disk space to a file. If you cannot predict the number of nodes, allow the default value to be applied. Specify a larger number if you expect to add nodes, but avoid wildly overestimating as this can affect buffer operations.

You can change the number of nodes using the **mmchfs** command. Changing this value affects storage pools created after the value was set; so, for example, if you need to increase this value on a storage pool, you could change the value, create a new storage pool, and migrate the data from one pool to the other.

Windows drive letter

In a Windows environment, you must associate a drive letter with a file system before it can be mounted. The drive letter can be specified and changed with the **-t** option of the **mmcrfs** and **mmchfs** commands. GPFS does not assign a default drive letter when one is not specified.

The number of available drive letters restricts the number of file systems that can be mounted on Windows.

Note: Certain applications give special meaning to drive letters **A:**, **B:**, and **C:**, which could cause problems if they are assigned to a GPFS file system.

Mountpoint directory

Every GPFS file system has a default mount point associated with it. This mount point can be specified and changed with the **-T** option of the **mmcrfs** and **mmchfs** commands.

If you do not specify a mount point when you create the file system, GPFS will set the default mount point to **/gpfs/***DeviceName*.

Assign mount command options

Options may be passed to the file system **mount** command using the **-o** option on the **mmchfs** command.

In particular, you can choose the option to perform quota activation automatically when a file system is mounted.

Enabling quotas

The GPFS quota system can help you control file system usage.

Quotas can be defined for individual users, groups of users, or filesets. Quotas can be set on the total number of files and the total amount of data space consumed. When setting quota limits for a file system, the system administrator should consider the replication factors of the file system. Quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system that has either data replication or metadata replication set to a value of two, the values reported on by both the **mmlsquota** and **mmrepquota** commands are double the value reported by the **ls** command.

Whether or not to enable quotas when a file system is mounted may be specified at file system creation by using the **-Q** option on the **mmcrfs** command or changed at a later time by using the **-Q** option on the **mmchfs** command. After the file system has been mounted, quota values are established by issuing the **mmedquota** command and activated by issuing the **mmquotaon** command. The default is to *not* have quotas activated.

GPFS levels are defined at three limits that you can explicitly set using the **mmedquota** and **mmdefedquota** commands:

Soft limit
: Defines levels of disk space and files below which the user, group of users, or fileset can safely operate.

  Specified in units of kilobytes (k or K), megabytes (m or M), or gigabytes (g or G). If no suffix is provided, the number is assumed to be in bytes.

Hard limit
: Defines the maximum amount of disk space and number of files the user, group of users, or fileset can accumulate.

  Specified in units of kilobytes (k or K), megabytes (m or M), or gigabytes (g or G). If no suffix is provided, the number is assumed to be in bytes.

Grace period
: Allows the user, group of users, or fileset to exceed the soft limit for a specified period of time. The default period is one week. If usage is not reduced to a level below the soft limit during that time, the quota system interprets the soft limit as the hard limit and no further allocation is allowed. The user, group of users, or fileset can reset this condition by reducing usage enough to fall below the soft limit; or the administrator can increase the quota levels using the **mmedquota** or **mmdefedquota**.

For implications of quotas for different protocols, see *Implications of quotas for different protocols* in *IBM Spectrum Scale: Administration and Programming Reference*.

Default quotas:

Applying default quotas provides all new users, groups of users, or filesets with established minimum quota limits. If default quota values are not enabled, new users, new groups, or new filesets have a quota value of zero, which establishes no limit to the amount of space that can be used.

Default quota limits can be set or changed only if the **-Q yes** option is in effect for the file system. To set default quotas at the fileset level, the **--perfileset-quota** option must also be in effect. The **-Q yes** and **--perfileset-quota** options are specified when creating a file system with the **mmcrfs** command or changing file system attributes with the **mmchfs** command. Use the **mmlsfs** command to display the current settings of these quota options. Default quotas may then be enabled by issuing the **mmdefquotaon** command. Default values are established by issuing the **mmdefedquota** command.

Quota system files:

The GPFS quota system maintains three separate files that contain data about usage and limits.

These files reside in the root directory of the GPFS file systems when quotas are enabled:

- **user.quota**
- **group.quota**
- **fileset.quota**

All three **.quota** files are:

- Built with the information provided in the **mmedquota** and **mmdefedquota** commands.
- Updated through normal allocation operations throughout the file system and when the **mmcheckquota** command is issued.
- Readable by the **mmlsquota** and **mmrepquota** commands.

The **.quota** files are read from the root directory when mounting a file system with quotas enabled. When these files are read, one of three possible actions take place:

- The files contain quota information and the user wants these files to be used.
- The files contain quota information, however, the user wants different files to be used.

  To specify the use of different files, the **mmcheckquota** command *must* be issued prior to the **mount** of the file system.
- The files do not contain quota information. In this case the **mount** fails and appropriate error messages are issued. For more information regarding **mount** failures, see *IBM Spectrum Scale: Problem Determination Guide*.

Enable DMAPI

Whether or not the file system can be monitored and managed by the GPFS Data Management API (DMAPI) may be specified at file system creation by using the **-z** option on the **mmcrfs** command or changed at a later time by using the **-z** option on the **mmchfs** command.

The default is *not* to enable DMAPI for the file system.

For more information about DMAPI for GPFS, see *IBM Spectrum Scale: Data Management API Guide*.

Verifying disk usage

The **-v** option controls whether the **mmcrfs** command checks whether the specified disks can safely be added to the file system.

The default (**-v yes**) is to perform the check and fail the command if any of the disks appear to belong to some other GPFS file system. You should override the default behavior and specify **-v no** only if the **mmcrfs** command rejects the disks and you are certain that *all* of the disks indeed do not belong to an active GPFS file system. An example for an appropriate use of **-v no** is the case where an **mmcrfs** command is interrupted for some reason and you are reissuing the command. Another example would be if you are reusing disks from an old GPFS file system that was not formally destroyed with the **mmdelfs** command.

Important: Using **mmcrfs -v no** on a disk that already belongs to a file system will corrupt that file system.

Changing the file system format to the latest level

You can change the file system format to the latest format supported by the currently-installed level of GPFS by issuing the **mmchfs** command with the **-V full** option or the **-V compat** option.

The **full** option enables all new functionality that requires different on-disk data structures. This may cause the file system to become permanently incompatible with earlier releases of GPFS. The **compat** option enables only changes that are backward compatible with the previous GPFS release. If all GPFS nodes that are accessing a file system (both local and remote) are running the latest level, then it is safe to use **full** option. Certain features may require you to run the **mmmigratefs** command to enable them.

Enabling file system features

By default, new file systems are created with all currently available features enabled.

Since this may prevent clusters that are running earlier GPFS releases from accessing the file system, you can enable only the file system features that are compatible with the specified release by issuing the **mmcrfs** command with the **--version** *Version* option. For more information, see *IBM Spectrum Scale: Administration and Programming Reference*.

Specifying whether the df command will report numbers based on quotas for the fileset

You can specify (when quotas are enforced for a fileset) whether the **df** command will report numbers based on the quotas for the fileset and not for the total file system.

To do so, use the **--filesetdf | --nofilesetdf** option on either the **mmchfs** command or the **mmcrfs** command.

Specifying the maximum number of files that can be created

The maximum number of files that can be created can be specified by using the **--inode-limit** option on the **mmcrfs** command and the **mmchfs** command.

Allowable values, which range from the current number of created inodes (determined by issuing the **mmdf** command with the **-F** option) through the maximum number of files that are supported, are constrained by the formula:

```
maximum number of files = (total file system space) / (inode size + subblock size)
```

You can determine the inode size (**-i**) and subblock size (value of the **-B** parameter / 32) of a file system by running the **mmlsfs** command. The maximum number of files in a file system may be specified at file system creation by using the **--inode-limit** option on the **mmcrfs** command, or it may be increased at a later time by using **--inode-limit** on the **mmchfs** command. This value defaults to the size of the file system at creation divided by 1 MB and cannot exceed the architectural limit. When a file system is created, 4084 inodes are used by default; these inodes are used by GPFS for internal system files.

For more information, see *mmcrfs command* and *mmchfs command* in *IBM Spectrum Scale: Administration and Programming Reference*.

The **--inode-limit** option applies only to the root fileset. When there are multiple inode spaces, use the **--inode-space** option of the **mmchfileset** command to alter the inode limits of independent filesets. The **mmchfileset** command can also be used to modify the root inode space. The **--inode-space** option of the **mmlsfs** command shows the sum of all inode spaces.

Inodes are allocated when they are used. When a file is deleted, the inode is reused, but inodes are never deallocated. When setting the maximum number of inodes in a file system, there is the option to preallocate inodes. However, in most cases there is no need to preallocate inodes because, by default,

inodes are allocated in sets as needed. If you do decide to preallocate inodes, be careful not to preallocate more inodes than will be used; otherwise, the allocated inodes will unnecessarily consume metadata space that cannot be reclaimed.

These options limit the maximum number of files that may actively exist within a file system. However, the maximum number of files in the file system may be restricted further by GPFS so the control structures associated with each file do not consume all of the file system space.

Further considerations when managing inodes:

1. For file systems that are supporting parallel file creates, as the total number of free inodes drops below 5% of the total number of inodes, there is the potential for slowdown in file system access. Take this into consideration when creating or changing your file system. Use the **mmdf** command to display the number of free inodes.
2. Excessively increasing the value for the maximum number of files may cause the allocation of too much disk space for control structures.

Controlling the order in which file systems are mounted

You can control the order in which the individual file systems are mounted at daemon startup (if **mmlsconfig autoload** shows yes) or when one of the **all** keywords is specified.

To do so, use the **--mount-priority** *Priority* option on the **mmcrfs**, **mmchfs**, or **mmremotefs** command.

The shared root file system for protocols must be mounted before other file systems that will be used to export protocol data.

A sample file system creation

To create a file system called **gpfs2** with the following properties:

- The disks for the file system that are listed in the file **/tmp/gpfs2dsk**
- Automatically mount the file system when the GPFS daemon starts (**-A yes**)
- Use a block size of 256 KB (**-B 256K**)
- Expect to mount it on 32 nodes (**-n 32**)
- Set both the default metadata replication and the maximum metadata replication to two (**-m 2 -M 2**)
- Set the default data replication to one and the maximum data replication to two (**-r 1 -R 2**)
- Use a default mount point of `/gpfs2` (**-T /gpfs2**)

Enter:

```
mmcrfs /dev/gpfs2 -F /tmp/gpfs2dsk -A yes -B 256K -n 32 -m 2 -M 2 -r 1 -R 2 -T /gpfs2
```

The system displays information similar to:

```
The following disks of gpfs2 will be formatted on node k194p03.tes.nnn.com:
    hd25n09: size 17796014 KB
    hd24n09: size 17796014 KB
    hd23n09: size 17796014 KB
Formatting file system ...
Disks up to size 59 GB can be added to storage pool system.
Creating Inode File
  56 % complete on Mon Mar  3 15:10:08 2014
 100 % complete on Mon Mar  3 15:10:11 2014
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
  44 % complete on Mon Mar  3 15:11:32 2014
  90 % complete on Mon Mar  3 15:11:37 2014
```

```
 100 % complete on Mon Mar  3 15:11:38 2014
Completed creation of file system /dev/gpfs2.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To confirm the file system configuration, issue the command:

```
mmlsfs gpfs2
```

The system displays information similar to:

```
flag                value                    description
------------------- ------------------------ -----------------------------------
 -f                 262144                   Minimum fragment size in bytes
 -i                 512                      Inode size in bytes
 -I                 32768                    Indirect block size in bytes
 -m                 2                        Default number of metadata replicas
 -M                 2                        Maximum number of metadata replicas
 -r                 1                        Default number of data replicas
 -R                 2                        Maximum number of data replicas
 -j                 scatter                  Block allocation type
 -D                 nfs4                     File locking semantics in effect
 -k                 all                      ACL semantics in effect
 -n                 32                       Estimated number of nodes that will mount file system
 -B                 262144                   Block size
 -Q                 none                     Quotas accounting enabled
                    none                     Quotas enforced
                    none                     Default quotas enabled
 --perfileset-quota yes                      Per-fileset quota enforcement
 --filesetdf        yes                      Fileset df enabled?
 -V                 14.20 (4.1.1.0)          File system version
 --create-time      Fri Jun 12 18:39:47 2015 File system creation time
 -z                 no                       Is DMAPI enabled?
 -L                 262144                   Logfile size
 -E                 yes                      Exact mtime mount option
 -S                 yes                      Suppress atime mount option
 -K                 whenpossible             Strict replica allocation option
 --fastea           yes                      Fast external attributes enabled?
 --encryption       no                       Encryption enabled?
 --inode-limit      2015232                  Maximum number of inodes
 --log-replicas     0                        Number of log replicas (max 2)
 --is4KAligned      yes                      is4KAligned?
 --rapid-repair     yes                      rapidRepair enabled?
 --write-cache-threshold 65536               HAWC Threshold (max 65536)
 -P                 system                   Disk storage pools in file system
 -d                 gpfs1001nsd;gpfs1002nsd  Disks in file system
 -A                 yes                      Automatic mount option
 -o                 none                     Additional mount options
 -T                 /gpfs2                   Default mount point
 --mount-priority   0                        Mount priority
```

For more information, see *mmcrfs command* and *mmlsfs command* in *IBM Spectrum Scale: Administration and Programming Reference*

Fileset considerations for creating protocol data exports

You can create exports on the entire file system, on sub-directories of a file system, or on filesets.

A fileset is a file system object that enables you to manage data at a finer granularity than the file system. You can perform administrative operations such as defining quotas, creating snapshots, and defining file placement policies and rules, and specifying inode space values at the fileset level, especially when the fileset is independent.

In IBM Spectrum Scale, you can create exports even without filesets. Depending on your data management strategy, choose either of the following ways to create exports:

Create exports on the entire file system or on sub-directories of the file system

In this option, the export represents a large space. You can create independent filesets over the directories in this space and have finer control over the export directory paths. Universities and organizations that require a departmental multi-tenancy solution can choose this option.

Review the following example to better understand this option.

As a storage administrator of an organization, you want to create separate storage space for every department and user of the organization:

1. Export the root directory of the file system.

   ```
   mmnfs export add /gpfs/fs0
   ```

   Note: You can create a sub-directory in the root directory and export it. For example: `mmnfs export add /gpfs/fs0/home`

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

2. Create filesets in the root directory:

   ```
   mmcrfileset fs0 hr_fileset --inode-space=new
   mmlinkfileset fs0 hr_fileset -J /gpfs/fs0/home/hr
   mmcrfileset fs0 user1_fileset --inode-space=new
   mmlinkfileset fs0 user1_fileset -J /gpfs/fs0/home/user1
   ```

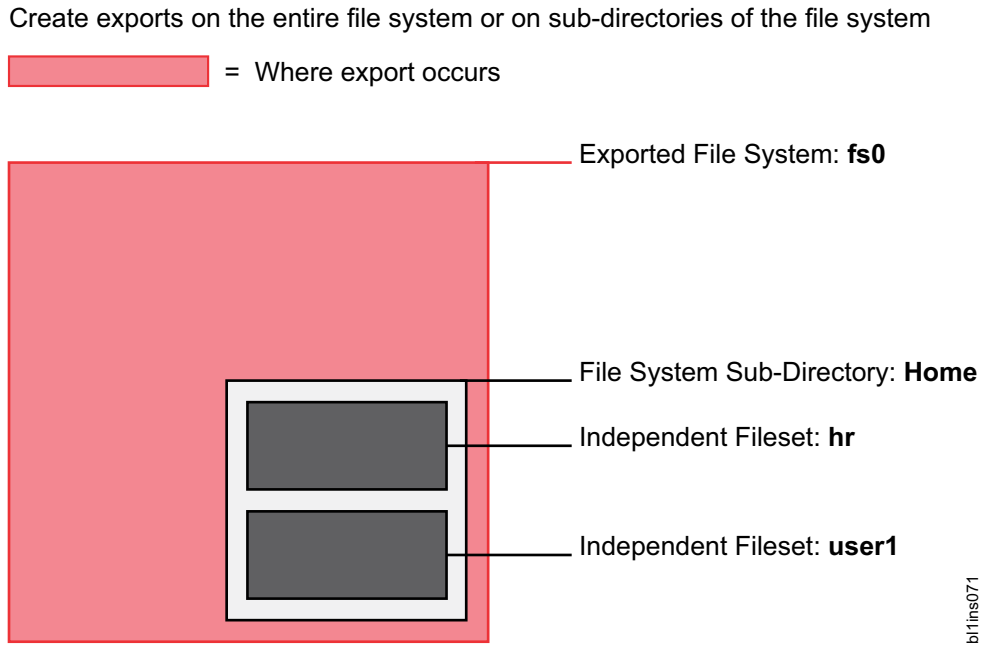
   For more information, see the following commands in the *IBM Spectrum Scale: Administration and Programming Reference*.

   - **mmcrfileset command**
   - **mmlinkfileset command**

3. Similarly, create independent filesets for other departments and users.

   You can assign quota to each user and department via the independent filesets.

   The NFS and SMB clients can now mount the export and access their directories.Access to the data in the export directories is controlled via the group and user ACLs. In this case, only the users who are in group 'HR', which has group ACLs to read/write into the 'hr' directory, can access the directory. The user user1 who is in the group 'group/HR' can perform read/write to the 'user1' directory and the 'hr' directory.



Create exports on independent filesets

In this option, the independent filesets represent discrete projects. One or more exports can be created on each fileset. You can apply the Information Lifecycle Management (ILM) policies over the filesets to automate the placement and management of file data. You can protect the export data by granting access permissions to specific workstations or IP addresses. Also, data in the exports can be preserved by the independent fileset's snapshot policy.

Review the following example to better understand this option.

You are a storage administrator of a private cloud hosting storage system that stores webcam data. You want to ensure that a webcam has access only to its storage directory. Also, you want the data analyzers to access all data so that they can look for activity and generate analytical reports.

1. Create an independent fileset 'web_cam_data'.

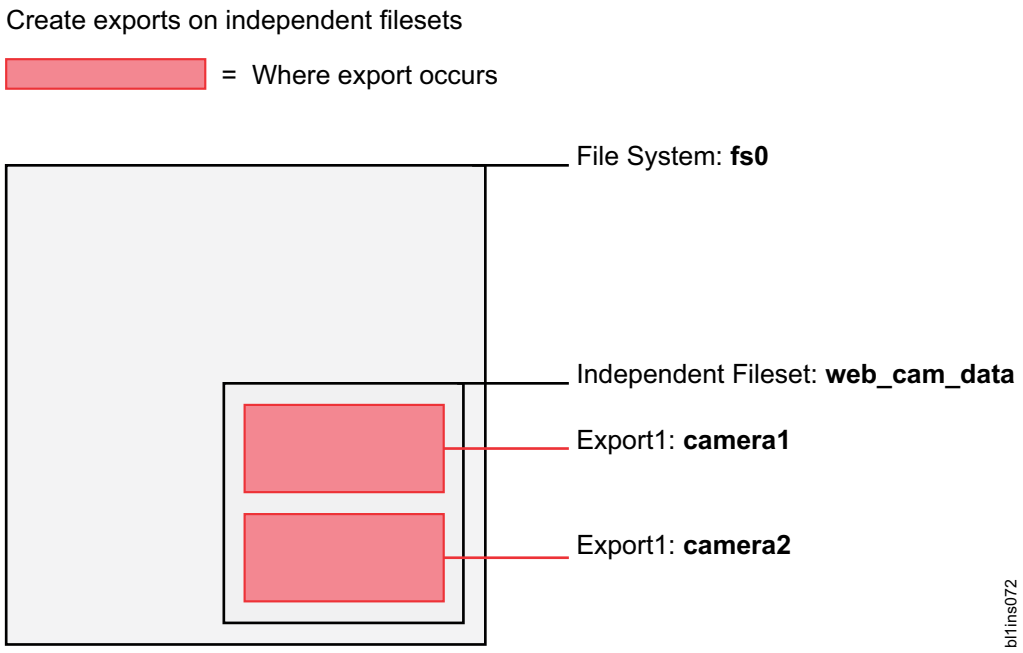
```
mmcrfileset fs0 web_cam_data --inode-space=new
mmlinkfileset fs0 web_cam_data -J /gpfs/fs0/web_cam_data
```

   Data from all webcams is stored in `/gpfs/fs0/web_cam_data`.

2. Create exports for both webcams.

```
mmnfs export add "/gpfs/fs0/web_cam_data/camera1" -c "9.3.101.2(Access_Type=RW);9.3.65.2(Access_Type=RO);9.3.65.3(Access_Type=RO)"
mmnfs export add "/gpfs/fs0/web_cam_data/camera2" -c "9.3.101.3(Access_Type=RW);9.3.65.2(Access_Type=RO);9.3.65.3(Access_Type=RO)"
```

The webcam1 (IP: 9.3.101.2) mounts and records data to the `camera1` export and the webcam2 (IP: 9.3.101.3) mounts and records data to the `camera2` export. The data analyzers (IP: 9.3.65.2 and 9.3.65.3) are given read access to both exports. Thus, the data is accessible only from the specified IP addresses.



Backup considerations for using Tivoli Storage Manager

If you are planning to use Tivoli Storage Manager to back up IBM Spectrum Scale file systems, refer to the following considerations.

Considerations for provisioning Tivoli Storage Manager servers to handle backup of IBM Spectrum Scale file systems

When backing up IBM Spectrum Scale file systems, the Tivoli Storage Manager server must be configured with adequate resources to handle a much larger load of client data, sessions, and mount points than typically encountered with single-user file systems such as workstations.

For cluster file system backup, a system architect must plan for multiple nodes (storage hosts) sending data in multiple streams, at high data rates to the server. Therefore, the Tivoli Storage Manager server must be equipped with the following:

- High bandwidth network connections to the IBM Spectrum Scale cluster
- Sufficient disk space organized into a Tivoli Storage Manager storage pool to be allocated to backup data from the cluster file system
- Sufficient number of mount points (or infinite for a disk class pool) accorded to the proxy client used for the cluster file system
- Sufficient number of Tivoli Storage Manager sessions permitted to the proxy client used for the cluster file system
- Sufficient number of physical tape drives to offload the backup data once it arrives into the disk pool

Tivoli Storage Manager data storage model

When using Tivoli Storage Manager for backing up IBM Spectrum Scale file systems, the Tivoli Storage Manager data storage architecture and its implications need to be considered.

In IBM Spectrum Scale, the following kinds of data are stored on disk:

- The file data that is the content stored in a file.
- The file metadata that includes all attributes related to the file. For example:

- Create, access, and modify times
- Size of the file, size occupied in file system, and number of blocks used
- Inode information, owner user id, owning group id, and mode bits
- POSIX rights or access control lists (ACLs)
- Flags to indicate whether the file is immutable or mutable, read only, or append only
- Extended attributes (EAs)

In Tivoli Storage Manager, the same file data and metadata is stored but the method of storing this data differs. The file content is stored in a TSM storage pool such as on disk or on tape while some of the metadata is stored in the Tivoli Storage Manager database. The primary reason for storing metadata in the Tivoli Storage Manager database is to provide fast access to information useful for backup requests.

However, not all metadata is stored in the Tivoli Storage Manager database. Access control lists (ACLs) and extended attributes (EAs) are stored with the file content in the storage pool (media depends on storage pool type). This has the following implications:

- When the ACL or EA of a file changes then the next backup job backs up the whole file again. This occurs because the file content, ACL, and EA are stored together in the TSM data pool, for example on tape and they need to be updated as one entity.

  Note: ACLs are inherited in the IBM Spectrum Scale file system. Therefore, an ACL on a top-level directory object can be inherited to all the descendant objects. ACL changes to a top-level directory object are therefore propagated down through the object tree hierarchy, rippling the change through all objects that inherited the original ACL. The number of files to be backed up increases even though nothing else in these files has changed. A surprisingly large backup workload can be induced by a seemingly innocent, small change to an ACL to a top level directory object.

  When Tivoli Storage Manager for Space Management (HSM) capability is enabled, an ACL change such as this occurs to objects that are currently migrated to offline storage as well. These files will then need to be recalled during the next backup cycle to enable the updated ACL to be stored with the file data once again in their TSM backup storage pool.
- Renaming of a file also leads to a backup of the whole file because the TSM database is indexed by file object path name.

You can use the following approaches to mitigate the size of a backup workload when widely inherited ACLs are likely to be changed frequently.

- Avoid renaming directories that are close to the file system root.
- Avoid ACL and EA changes in migrated files as much as possible.
- Consider using the `skipacl` or `skipaclupdatecheck` options of the Tivoli Storage Manager client.

  Important: Be certain to note the implications of using these options by referring to *Clients options reference* in the *Backup-archive Client options and commands* section of the Tivoli Storage Manager documentation set on IBM Knowledge Center.

  Note: Using the `skipacl` option also omits EAs from the backup data store in the TSM backup pool. Using this option can be considered when static ACL structures are used that can be reestablished through another tool or operation external to the Tivoli® Storage Manager restore operation. If you are using this approach, ensure that the ACL is restored either manually or automatically, by inheritance, to avoid an unauthorized user getting access to a file or a directory after it is restored.

How to identify backup and migration candidates

When using Tivoli Storage Manager for backing up IBM Spectrum Scale file systems, there are several choices for the method used to identify backup and migration candidates.

Limitations when using TSM Backup-Archive client to identify backup candidates

Using TSM Backup-Archive client to traverse IBM Spectrum Scale file systems to identify backup candidates does not scale well. For this reason, using the IBM Spectrum Scale **mmapplypolicy** engine is preferable because it is much faster to scan the file system for identifying backup candidates than traversing the file system.

Therefore, for processing backups on larger file systems, use the IBM Spectrum Scale command **mmbackup** instead of using the TSM Backup-Archive client commands such as **dsmc expire** or **dsmc selective** or **dsmc incremental** directly. Using the **mmbackup** command also provides the following benefits:

- Backup activities are run in parallel by using multiple IBM Spectrum Scale cluster nodes to send backup data in parallel to the Tivoli Storage Manager server.
- **mmbackup** creates a local shadow of the Tivoli Storage Manager database in the file system and uses it along with the policy engine to identify candidate files for backup. The Tivoli Storage Manager server does not need to be queried for this information saving time when calculating the backup candidate list.
  - **mmbackup** and its use of the policy engine can select candidates faster than the **dsmc** progressive incremental operation that is bounded by walk of the file system using the POSIX directory and file status reading functions.
  - Using **dsmc selective** with lists generated by **mmbackup** is also faster than using **dsmc incremental** even with similar lists generated by **mmbackup**.

  Note: It is recommended that scheduled backups of an IBM Spectrum Scale file system use **mmbackup** because **mmbackup** does not actively query the Tivoli Storage Manager server to calculate backup candidates. However, events such as file space deletion or file deletion executed on Tivoli Storage Manager server are not recognized until the user triggers a synchronization between the **mmbackup** shadow database and the TSM database.

The following table contains a detailed comparison of **mmbackup** and TSM Backup-Archive client backup commands:

Table 6. Comparison of **mmbackup** *and TSM Backup-Archive client backup commands*

	IBM Spectrum Scale policy-driven backup (mmbackup)	**Tivoli Storage Manager progressive incremental backup (dsmc incremental)**
Detects changes in files and sends a new copy of the file to the server.	Yes	Yes
Detects changes in metadata and updates the file metadata on the server or sends a new copy of the file to the server (for ACL/EA changes).	Yes	Yes
Detects directory move, copy, or rename functions, and sends a new copy of the file to the server.	Yes	Yes
Detects local file deletion and expires the file on the server.	Yes	Yes
Detects Tivoli Storage Manager file space deletion or node/policy changes, and sends a new copy of the file to the server.	No*	Yes
Detects file deletion from the Tivoli Storage Manager server and sends a new copy of the file to the server.	No*	Yes

Table 6. Comparison of **mmbackup** *and TSM Backup-Archive client backup commands (continued)*

	IBM Spectrum Scale policy-driven backup (mmbackup)	**Tivoli Storage Manager progressive incremental backup (dsmc incremental)**
Detects additions of new exclude rules and expires the file on the server.	Yes	Yes
Detects policy changes made to include rules and rebinds the file to the new storage pool.	No**	Yes
Detects copy mode and copy frequency configuration changes.	No*	Yes
Detects migration state changes (Tivoli Storage Manager for Space Management) and updates the server object.	Yes	Yes
Detects that a file wasn't processed successfully during a backup operation and attempts again at the next backup.	Yes	Yes
Supports TSM Virtual Mount Points to divide a file system into smaller segments to reduce database size and contention.	No	Yes
* The **mmbackup** command queries the Tivoli Storage Manager server only once at the time of the first backup. Changes that are performed on the Tivoli Storage Manager server by using the Tivoli Storage Manager administrative client cannot be detected by **mmbackup** processing. You must rebuild the **mmbackup** shadow database if the Tivoli Storage Manager server file space changes.		
** Tivoli Storage Manager includes rules with associated management class bindings that cannot be detected by **mmbackup** processing. Therefore, **mmbackup** processing does not rebind a file if a management class changes include rules.		

If you use TSM Backup-Archive client backup commands on file systems that are otherwise handled by using **mmbackup**, the shadow database maintained by **mmbackup** loses its synchronization with the TSM inventory. In such cases, you need to resynchronize with the Tivoli Storage Manager server which will inform **mmbackup** of the recent backup activities conducted with the **dsmc** command. Resynchronization might be a very time-consuming activity for large file systems with a high number of backed up items. To avoid these scenarios, use the **mmbackup** command only.

If you have used **dsmc selective** or **dsmc incremental** since starting to use **mmbackup** and need to manually trigger a synchronization between the **mmbackup** maintained shadow database and the Tivoli Storage Manager server:

- Use the **mmbackup --rebuild** if you need to do a synchronization only.
- Use the **mmbackup -q** if you need to do a synchronization followed by a backup of the corresponding file system.

Using the Tivoli Storage Manager for Space Management client to identify migration candidates

Using Tivoli Storage Manager for Space Management clients for traversing IBM Spectrum Scale file system to identify migration candidates does not scale well. The Tivoli Storage Manager **automigration** daemons consume space in the file system and also consume CPU resources. They do not have access to

the internal structures of the file system in the way that the IBM Spectrum Scale **mmapplypolicy** command does, and so they cannot scale. Use the following steps instead:

1. Set the following environment variable before the installation of the Tivoli Storage Manager for Space Management client to prevent the **automigration** daemons from starting during the installation:

   ```
   export HSMINSTALLMODE=SCOUTFREE
   ```

   It is recommended to place this setting into the profile file of the root user.
2. Add the following option to your Tivoli Storage Manager client configuration file, `dsm.opt`, to prevent the **automigration** daemons from starting after every system reboot:

   ```
   HSMDISABLEAUTOMIGDAEMONS YES
   ```
3. Add the following option to your Tivoli Storage Manager client configuration file, `dsm.opt`, to ensure that the object ID is added to the inode, so that the file list based reconciliation (two-way-orphan-check) can be used:

   ```
   HSMEXTOBjidattr YES
   ```
4. While the **automigration** daemons are disabled, changes such as removal of migrated files are not automatically propagated to the Tivoli Storage Manager server. For housekeeping purposes, you must run the Tivoli Storage Manager reconciliation either manually or in a scheduled manner. For more information, see *Reconciling by using a GPFS policy* in the Tivoli Storage Manager for Space Management documentation on IBM Knowledge Center.

Comparison of snapshot based backups and backups from live system

Backing up large file systems can take many hours or even days. When using the IBM Spectrum Scale command **mmbackup**, time is needed for the following steps.

- Scanning the system to identify the objects that need to be backed up or expired.
- Expiring all objects removed from the system.
- Backing up all new or changed objects.

If the backup is run on the live file system while it is active, objects selected for the backup job can be backed up at different points in time. This can lead to issues when temporary or transient files that were present during the scan time are removed by the time the backup command tries to send them to the Tivoli Storage Manager server. The attempt to back up a file that is removed fails and the need to back this object up is still recorded in the shadow database.

Instead of backing up from a live file system, an alternative is to use snapshot based backups. Using the snapshot adds additional actions of creating or reusing a snapshot and removing it when the overall backup process completes. However, this approach provides several advantages because a snapshot is a point in time view of the file system that is read-only and it can be used for backing up the file system for as long as is necessary to complete the backup job. The advantages of following this approach are as follows:

- Transient or temporary files are backed up, provided they existed at the time the snapshot was taken.
- Protection against failures to back up due to server-side faults such as the Tivoli Storage Manager server running out of space. For example, if the database or storage pool becomes full or if the Tivoli Storage Manager server crashes, etc. In this case, a retry of the backup is possible for the point in time when the snapshot has been taken with no loss of function or backup.
- Retention of a backup within the system. Snapshots can be kept for a period of time providing an online backup copy of all files. This can protect against accidental deletions or modifications, and can be used to retrieve an earlier version of a file, etc.
- A means to fulfill a data protection policy even if the backup activity to Tivoli Storage Manager exceeds the nominal time window allotted. The snapshot can be kept for several days until backups are complete, and multiple snapshots can be kept until backup completes for each of them.

IBM Spectrum Scale provides the capability to create snapshots for a complete file system, known as global snapshots, and for independent filesets, known as fileset snapshots.

While snapshot based backups provide several advantages, the following considerations apply when using the snapshot capability:

- Snapshots might consume space; usually a snapshot used for backup is removed shortly after the backup operation finishes. Long lived snapshots retain their copy of the data blocks, taken from the file system's pool of free blocks. As they age, the snapshots consume more data blocks owing to the changes made in the read or write view of the file system.
- Snapshot deletion can take time depending on the number of changes that need to be handled while removing the snapshot. In general, the older a snapshot is, the more work it will require to delete it.
- Special consideration for use of Tivoli Storage Manager for Space Management:
  - When a migrated file stub that is already part of a snapshot is removed, a recall is initiated to keep the snapshot consistent. This is required because removal of the stub invalidates the offline references to the stored data. The recall is to fill blocks on disk and assign them to the snapshot view. Once the stub is removed from the file system and a reconcile process removes this file from the Space Management pool on the Tivoli Storage Manager server, there are no longer any references to the file data except the snapshot copy.

File system and fileset backups with Tivoli Storage Manager

Starting with IBM Spectrum Scale 4.1.1, **mmbackup** supports backup from independent filesets in addition to backup of the whole file system. Independent filesets have similar capabilities as a file system in terms of quota management, dependent filesets, snapshots, and having their own inode space.

Fileset **mmbackup** provides a finer granularity for backup purposes and permits the administrator to:

- Have different backup schedules for different filesets. For example, providing the flexibility that filesets containing more important data to be backed up more often than other filesets.
- Have a fileset dedicated for temporary or transient files or files that do not need to be backed up.
- Use file system backups in conjunction with fileset backup to implement a dual data protection scheme such that file system backup goes to one Tivoli Storage Manager server while fileset backup goes to a different Tivoli Storage Manager server.

Backups of file systems and independent filesets are controlled by using the **mmbackup** option `--scope`.

In the examples used in this topic, the following environment is assumed:

```
tsm server name             => tsm1,tsm2,...
file system device name     => gpfs0 or /dev/gpfs0
file system mountpoint      => /gpfs0

fileset names/junction path
---------------------------
depfset1,depfset2,...     /gpfs0/depfset1,/gpfs0/depfset2         for dependent filesets
indepfset1,indepfset2,... /gpfs0/indepfset1,/ibm/gpfs0/indepfset2 for indepdendent filesets
```

File system backup

File system backup protects the whole file system. The default value for option `--scope` is `filesystem` and thus it can be omitted.

To backup the `gpfs0` file system in its entirety to the TSM server named `tsm1`, use the following command:

```
mmbackup gpfs0 --tsm-servers tsm1
mmbackup gpfs0 --tsm-servers tsm1 --scope filesystem
mmbackup /dev/gpfs0 --tsm-servers tsm1
mmbackup /dev/gpfs0 --tsm-servers tsm1 --scope filesystem
mmbackup /gpfs0 --tsm-servers tsm1
mmbackup /gpfs0 --tsm-servers tsm1 --scope filesystem
```

This example shows the file system backup of `gpfs0` using the short or long device name (first 4 lines) or the mount point as a directory input (last 2 lines).

Independent fileset backup

Independent fileset backup protects all files that belong to the inode space of a single independent fileset. The backup of an independent fileset might include other dependent filesets and folders, but not nested independent filesets because each independent fileset has its own inode space.

The following information describes the **mmbackup** capability to support independent fileset. The examples with TSM show the potential/likely use of this capability.

Note: The current version of TSM does not support independent fileset backup, and this information does not imply that it will be available in a TSM release. This information is only provided as an illustration of independent fileset backup capability if it is introduced in a product that can exploit it.

To backup the independent fileset `indepfset1` of the file system `gpfs0` to Tivoli Storage Manager server named `tsm1`, use the following command:

```
mmbackup /gpfs0/indepfset1 --tsm-servers tsm1 --scope inodespace
```

Fileset backup is an additional option to existing IBM Spectrum Scale backup or data protection options. Therefore, starting with a file system in which the whole file system has already been backed up on a regular basis, administrators can also start protecting data with fileset backup on some or all filesets. Administrators may choose to utilize fileset backup at any time, with no limit. It is not required to back up the whole file system in either one mode or another unless the chosen data protection approach requires it. Also, administrators can choose different Tivoli Storage Manager servers for some or all fileset backups if it is desired to have these on separate servers. For example, a valid configuration may have one Tivoli Storage Manager server that only gets whole file system backups, and a different one for fileset backups.

Note: When mixing file system and fileset backup on the same TSM server, consider that a target file that is handled by both backup approaches gets backed up twice. Each backup activity consumes one storage version of the file. Thus, the same file version is stored twice on the Tivoli Storage Manager server.

If migrating to use only fileset backup to protect the whole file systems, ensure that all independent filesets are backed up and keep the backup of all filesets current. Remember to include a backup of the `root` fileset as well by using a command such as:

```
mmbackup /gpfs/gpfs0 --scope inode-space --tsm-servers tsm1
```

To verify the completeness of the data protection using fileset backup only, use the following steps:

1. Identify all independent filesets available in the file system by using the following command:

   `mmlsfileset` *`device`* `-L`

   Identify the independent filesets by the InodeSpace column. The value identifies the corresponding inode space or independent fileset while the first occurrence refers to the corresponding fileset name and fileset path.

   Note: Ensure that a backup is maintained of the fileset called `root` that is created when the file system is created and that can be seen as the first fileset of the file system.
2. For every identified independent fileset, verify that a shadow database exists as follows:
   a. Check for the file `.mmbackupShadow.<digit>.<TSMserverName>.fileset` in the fileset junction directory.
   b. If the file exists, determine the time of last backup by looking at the header line of this file. The backup date is stored as last value of this line.

      For example:

```
head -n1 /gpfs0/indepfset1/.mmbackupShadow.*.fileset
%%mshadow0%%:00_BACKUP_FILES_41:1400:/gpfs0:mmbackup:1:Mon Apr 20 13:48:45 2015
```

Where *Mon Apr 20 13:48:45 2015* in this example is the time of last backup taken for this fileset.

3. If any independent filesets are missing their corresponding `.mmbackupShadow.*` files, or if they exist but are older than the data protection limit for their backup time, then start or schedule a backup of these filesets.

   Note: This action needs to be done for every file system for which the fileset backup approach is chosen.

Note: Backup of a nested independent fileset is not supported. To work around this limitation, unlink the nested fileset and link it at another location in the root fileset prior to beginning to use fileset backup pervasively.

Considerations for using fileset backup with Tivoli Storage Manager

Tivoli Storage Manager stores backup data indexed by file or directory object path names in its database. However, file system objects are identified for HSM by their object ID extended attributes. In both cases, Tivoli Storage Manager is not aware of the fileset entity. Therefore, the fileset configuration of an IBM Spectrum Scale cluster is neither backed up nor can it be restored by Tivoli Storage Manager alone, unless separate action is taken.

IBM Spectrum Scale does provide commands such as **mmbackupconfig** and **mmrestoreconfig** to backup and restore file system configuration data.

Even though backups of independent filesets may be stored on a plurality of Tivoli Storage Manager servers, Tivoli Storage Manager for Space Management can only utilize one server per managed file system if **mmbackup** is used on that file system and sends the backup data to that same server. Tivoli Storage Manager HSM multiserver feature is not integrated with the **mmbackup** command.

If space management is enabled for a IBM Spectrum Scale file system, and a plurality of Tivoli Storage Manager servers are utilized for backup, the following limitations apply to the use of Tivoli Storage Manager for Space Management:

- The setting of `migrequiresbkup` cannot be utilized since the server for space management might not have access to the database for the server doing backup on a particular object.
- The *inline copy* feature that allows the backup of a migrated object through copying data internally from the HSM pool to the backup pool inside the Tivoli Storage Manager server cannot be utilized since migrated data may reside on a different server than the one performing the backup.

The following special considerations apply when using fileset backup with Tivoli Storage Manager:

- TSM Backup-Archive client does not restore fileset configuration information. Therefore, a full file system backup is not sufficient for disaster recovery when the whole file system structure needs to be recovered.
- Nested independent filesets cannot be backed up when using the fileset backups. Resolve the nesting before starting to protect the data with **mmbackup** on each fileset.
- The following operations are discouraged: fileset unlink, fileset junction path change via unlink and link operations, fileset deletion. These operations will incur a heavy load on the Tivoli Storage Manager server at the next backup activity due to the significant alteration of the path name space that result from these operations. If a fileset must be moved in the file system, be prepared for the extra time that might be required to run the next backup.
- If only utilizing fileset backup, be sure to include every fileset, including the root fileset to ensure complete data protection.
- When both file system and fileset backups use the same Tivoli Storage Manager server, new and changed files may be unintentionally backed up twice. This consumes two of the file versions stored in the Tivoli Storage Manager server with the same file data.

Loss of data protection may occur if a fileset is unlinked or deleted for an extended period of time and the objects formerly contained in that fileset are allowed to expire from the backup sets.

- Do not move, rename, or relink a fileset, unless a new full backup of the fileset is expected to be made immediately after the restructuring occurs. Tivoli Storage Manager server only recognizes objects in backup only as long as they remain in their original path name from the root of the file tree (mount point).

The following special considerations apply when using fileset backup and Tivoli Storage Manager for Space Management:

- When using fileset backup to different Tivoli Storage Manager servers, the server setting **migrequiresbkup** must be set to NO to be able to migrate files.
- Automatic server selection for backup by **mmbackup** is not supported and therefore the Tivoli Storage Manager *multiple HSM server* feature is not integrated with **mmbackup** data protection of IBM Spectrum Scale. Using fileset backup to different Tivoli Storage Manager servers in conjunction with the *multiple HSM server* feature requires special handling and configuration because backup and space management rules must match so that files are handled by the same Tivoli Storage Manager server.

Considerations for backing up file systems that are HSM space managed with Tivoli Storage Manager for Space Management

When Tivoli Storage Manager for Space Management is used to migrate data to secondary storage pools (for example, tape volumes), this migrated data might need to be recalled to disk if the file becomes a backup candidate due to certain changes a user may apply to it.

Recall will automatically occur synchronously if the user attempts to change the file data. However, no synchronous recall occurs in the following scenarios:

- If a user changes the owner, group, mode, access control list, or extended attributes of the file.
- If the user renames the file or any of the parent directories of the file.

Any of these changes, however does make the file a candidate for the next **mmbackup** invocation. When **mmbackup** supplies the path to a migrated file to Tivoli Storage Manager for backup, synchronous recall will occur. This can lead to a "Recall Storm" in which tape library and Tivoli Storage Manager server resources become overburdened handling many simultaneous recall requests. Disk space is also consumed immediately by recalled data. These are undesirable outcomes and can interfere with the normal processing of the customer's workload.

Since **mmbackup** is able to discern that the data was migrated for such files, it will defer the backup, and instead append a record referring to the file's path name into a special file in the root of the file system, or fileset in the case of fileset-level backup. This list can then be used by the system administrator to schedule recall of the deferred files data to permit the backup to occur on the next invocation of the **mmbackup** command. By deferring the recall to the system administrator, **mmbackup** prevents unexpected, excessive tape library activity which might occur if many such migrated files were nominated for backup due to user actions such as renaming a high level directory or changing owner, group, or modes on many files.

The system administrator must schedule the recall for the migrated objects omitted by **mmbackup** according to the availability of tape and disk resources in the cluster. For information about optimized tape recall, see *Optimized tape recall processing* in the Tivoli Storage Manager for Space Management documentation in IBM Knowledge Center.

IBM Spectrum Scale license designation

According to the IBM Spectrum Scale Licensing Agreement, each node in the cluster must be designated as possessing an IBM Spectrum Scale Client license, an IBM Spectrum Scale FPO license, or an IBM Spectrum Scale Server license.

The full text of the Licensing Agreement is provided with the installation media and can be found at the IBM Software license agreements website (www.ibm.com/software/sla/sladb.nsf).

The type of license that is associated with any one node depends on the functional roles that the node has been designated to perform.

IBM Spectrum Scale Client license
The IBM Spectrum Scale Client license permits exchange of data between nodes that locally mount the same GPFS file system. No other export of the data is permitted. The GPFS client may not be used for nodes to share GPFS data directly through any application, service, protocol or method, such as Network File System (NFS), Common Internet File System (CIFS), File Transfer Protocol (FTP), or Hypertext Transfer Protocol (HTTP). For these functions, an IBM Spectrum Scale Server license would be required.

IBM Spectrum Scale FPO license
The IBM Spectrum Scale FPO license permits the licensed node to perform NSD server functions for sharing GPFS data with other nodes that have an IBM Spectrum Scale FPO or IBM Spectrum Scale Server license. This license cannot be used to share data with nodes that have an IBM Spectrum Scale Client license or non-GPFS nodes.

IBM Spectrum Scale server license
The IBM Spectrum Scale Server license permits the licensed node to perform GPFS management functions such as cluster configuration manager, quorum node, manager node, and Network Shared Disk (NSD) server. In addition, the IBM Spectrum Scale Server license permits the licensed node to share GPFS data directly through any application, service protocol or method such as NFS, CIFS, FTP, or HTTP.

These licenses are all valid for use in the IBM Spectrum Scale Express Edition, IBM Spectrum Scale Standard Edition, and IBM Spectrum Scale Advanced Edition.

The IBM Spectrum Scale license designation is achieved by issuing the appropriate **mmchlicense** command. The number and type of licenses currently in effect for the cluster can be viewed using the **mmlslicense** command. For more information about these commands, see *IBM Spectrum Scale: Administration and Programming Reference*.

Planning for Protocols

Authentication considerations

To enable read and write access to directories and files for the users on the IBM Spectrum Scale system, you must configure user authentication on the system. Only one user authentication method, and only one instance of that method, can be supported.

The following authentication services can be configured with the IBM Spectrum Scale system for file protocol access:

- Microsoft Active Directory (AD)
- Lightweight Directory Access Protocol (LDAP)
- Network Information Service (NIS) for NFS client access
- User defined

The following authentication services can be configured with the IBM Spectrum Scale system for object access:

- Microsoft Active Directory (AD)
- Lightweight Directory Access Protocol (LDAP)
- Local authentication

- User defined

The following matrix gives a quick overview of the supported authentication configurations for both file and object access.

- ✓: Supported
- X: Not supported
- NA: Not applicable

Table 7. Authentication support matrix

Authentication method	ID mapping method	SMB	SMB with Kerberos	NFSV3	NFSV3 with Kerberos	NFSV4	NFSV4 with Kerberos	Object
User-defined	User-defined	NA	NA	NA	NA	NA	NA	NA
LDAP with TLS	LDAP	✓	NA	✓	NA	✓	NA	✓
LDAP with Kerberos	LDAP	✓	✓	✓	✓	✓	✓	NA
LDAP with Kerberos and TLS	LDAP	✓	✓	✓	✓	✓	✓	NA
LDAP without TLS and without Kerberos	LDAP	✓	NA	✓	NA	✓	NA	✓
AD	Automatic	✓	✓	X	X	X	X	✓
AD	RFC2307	✓	✓	✓	✓	✓	✓	✓
AD	LDAP	✓	✓	✓	X	X	X	✓
NIS	NIS	NA	NA	✓	NA	✓	NA	NA
Local	None	NA	NA	NA	NA	NA	NA	✓

Note:

- The ID mapping option that is given in this table is only applicable for file access. Ignore the ID mapping details that are listed in the table if you are looking for the supported configurations for object access.
- In the User-defined mode, the customer is free to choose the authentication and ID mapping methods for file and object and manage on their own. That is, the authentication needs to be configured by the administrator outside of the IBM Spectrum Scale commands and ensure that it is common and consistent across the cluster.
- If LDAP-based authentication is used, ACL management for SMB is not supported.

The following diagram shows the high-level overview of the authentication configuration.

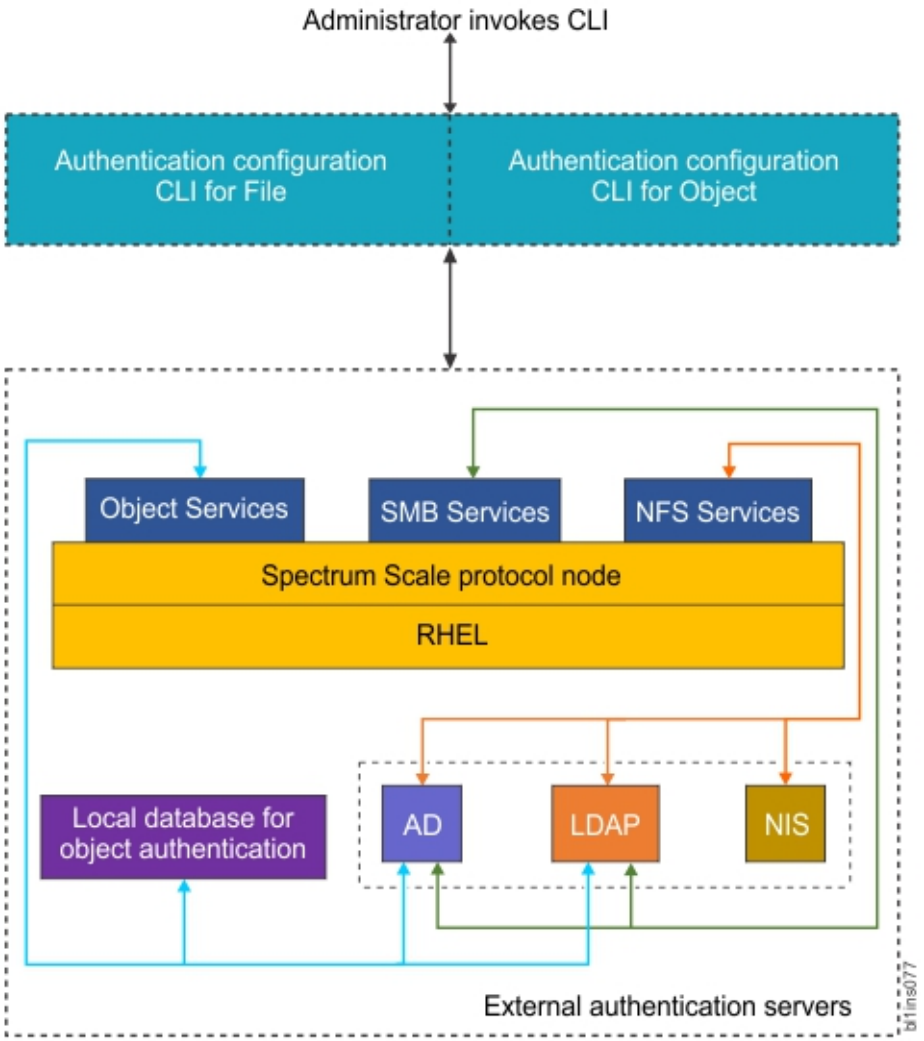


Figure 8. High-level overview of protocol user authentication

The authentication requests that are received from the client systems are handled by the corresponding services in the IBM Spectrum Scale system. For example, if a user needs to access the NFS data, the NFS services resolves the access request by interacting with the corresponding authentication and ID mapping servers.

Impact of enabling and disabling protocols on authentication

The following are the recommendations for enabling and disabling protocols.

- If authentication is already configured in the system, then the protocols that are enabled in the system to access data cannot be disabled, unless the system administrator cleans up the authentication configuration.
- If authentication is configured, a disabled protocol cannot be enabled. Hence, it is vital to plan properly to decide the protocols that you want to avail now and in near future and enable them correctly in advance.
- If authentication method chosen is AD, then authentication cannot be configured unless the SMB protocol is enabled and started.
- Protocols cannot be disabled unless authentication is unconfigured. Disabling of a protocol is a critical action and it might result in loss of access to the data hosted by that protocol or other protocols. After disabling the protocol, the authentication can be reconfigured with the same authentication servers and types so that the other enabled protocols can start functioning again.

Note: Instead of disabling protocols, the system administrators might stop a protocol if they do not need it. Stopping a protocol is the recommended action instead of disabling the protocol. Stopping a protocol is as good as removing the protocol from the cluster but it does not have an impact on the access to the data. Users can retain the access when the protocol is started again. Use the **mmces stop -nfs -a** command to stop a protocol on the CES nodes. For more information on stopping protocols, see For more details on these options, see the *mmces command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

- The restrictions for enabling or disabling protocols are not applicable if user-defined authentication method is used for file or object access.
- The authentication configuration needs to be removed to enable a new protocol in the system. If other protocols are already enabled in the system, the system administrator must reconfigure the authentication with the same authentication servers and types so that the other enabled protocols can start functioning again.

You can also remove ID mappings, along with authentication, if you want to completely remove the authentication configuration. This results in permanent loss of access to the data.

Authentication for file access

The system supports an external authentication service to authenticate users on the system. Before you configure authentication method, ensure that the external authentication service is set up correctly.

The following steps are involved in the user authentication for file access:

1. User tries to connect to the IBM Spectrum Scale system by using their credentials.
2. The IBM Spectrum Scale system contacts the authentication server to validate the user.
3. The IBM Spectrum Scale system contacts the ID map server that provides UIDs and GIDs of the user and user group to verify the identity of the user.
4. If the user credentials are valid, the user gains access to the system.

ID mapping

The authentication of the user or groups of users is also associated with the identification of their unique identifiers. To support data access to Microsoft Windows clients (SMB protocol) and to allow interoperability, that is, to share data among UNIX and Windows clients (SMB and NFS protocols), the IBM Spectrum Scale system must map Windows SID to UNIX UID/GID. This process is referred to as ID mapping and the map is referred to as ID map. The ID mapping can be done either internally in the IBM Spectrum Scale system or in an external authentication server.

ID mapping is part of the user identification process in user authentication. The purpose of identification is to identify users and infrastructure components. Identification methods include unique user identifiers (IDs), keys, or fingerprints such as a public Secure Shell (SSH) key, and digital certificates such as a certificate of a web server.

UNIX based system such as the IBM Spectrum Scale system use user names and user identifiers (UIDs) to represent users of the system. The user name is typically a human-readable sequence of alphanumeric characters and the UID is a positive integer value. When a user logs on to a UNIX system, the operating system looks up the UID and then uses this UID for further representation of the user. User names, UIDs, and the mapping of user names to UIDs are stored locally in the `/etc/passwd` file or on an external directory service such as Active Directory (AD), Lightweight Directory Access Protocol (LDAP), Keystone, or Network Information Service (NIS).

UNIX systems implement groups to maintain sets of users that have the same group permissions to access certain system resources. Similar to user names and UIDs, a UNIX system also maintains group names and group identifiers (GID). A UNIX user can be a member of one or more groups, where one group is the primary or default group. Group names, GIDs, the mapping of group names to GIDs, and

the memberships of users to groups are stored locally in the /etc/group file or on an external directory service such as Active Directory, LDAP, Keystone, or NIS. The primary group of a user is stored in /etc/passwd or in an external directory service.

Windows systems reference all operating system entities as resources. For example, users, groups, computers, and so on. Each resource is represented by a security identifier (SID). Resource names and SIDs are stored locally in the Windows registry or in an external directory service such as Active Directory or LDAP. The following methods are used to map Windows SID to UNIX UID and GID:

- External ID mapping methods
  - RFC2307 when AD-based authentication is used
  - LDAP when LDAP-based authentication is used
- Internal ID mapping method
  - Automatic ID mapping when AD-based authentication is used

External ID mapping

A UID or GID of a user or group is created and stored in an external server such as Microsoft Active Directory, NIS server, or LDAP server.

External ID mapping is useful when user UID or group GID is preexisting in the environment. For example, if NFS client with UID and GID as 100 exists in the environment, and you want a certain share to be accessed by both SMB and NFS client, then you can use an external ID mapping server, assign UID/GID 100 to the SMB user, and thus, allow both SMB and NFS client to access same data.

Note: The external server administrator is responsible for creating or populating the UID/GID for the user/group in their respective servers.

The IBM Spectrum Scale system supports the following servers for external ID mapping:

- LDAP server, where the UID or GID is stored in a dedicated field in the user or group object on the LDAP server.
- AD server with RFC2307 schema extension defined. The UID or GID of a user or group that is defined in AD server is stored in a dedicated field of the user or group object.

The UID/GID defined in external server can be used by the IBM Spectrum Scale system.

LDAP ID mapping is supported only when the IBM Spectrum Scale is configured with LDAP authentication.

Internal ID mapping

UID or GID of a user or group is created automatically by the IBM Spectrum Scale system and stored in the internal repositories.

When external ID mapping server is not present in the environment or cannot be used, the IBM Spectrum Scale system uses its internal ID mapping method to create the UID/GID.

IBM Spectrum Scale supports Automatic ID mapping method if AD-based authentication is used. Automatic ID mapping method uses a reserved ID range to allocate ID based on the following logic. A user or group in AD is identified by SID, which includes a component that is called RID. Whenever a user or group from an AD domain accesses IBM Spectrum Scale, a range is allocated per AD domain. UID or GID is then allocated depending upon this range and the RID of the user/group.

Internal ID mapping cannot be used when the user UID or group GID is preexisting in the environment. However, while using internal ID mapping and if an NFS client wants to access data, then the NFS client

must have UID or GID that is identical to the one created by the IBM Spectrum Scale system.

Other supported authentication elements for file access

The system supports the following authentication elements as well:

- **Netgroups:** Groups of hosts are used to restrict access for mounting NFS exports on a set of hosts, and deny mounting on the remainder of the hosts. The IBM Spectrum Scale system supports only the netgroups that are stored in NIS and in Lightweight Directory Access Protocol (LDAP).
- **Kerberos:** Kerberos is a network authentication protocol that provides secured communication by ensuring passwords are not sent over the network to the system. The system supports Kerberos with both AD and LDAP-based authentication. When you configure AD-based authentication, the Kerberos is enabled for the SMB access by default and Kerberos for NFS access is not supported.

  Kerberos is optional for both SMB and NFS access in LDAP. To enable Kerberos with LDAP, you need to integrate the system with MIT KDC
- **Transport Level Security (TLS):** The TLS protocol is primarily used to increase the security and integrity of data that is sent over the network. These protocols are based on public key cryptography and use digital certificates based on X.509 for identification.

Caching user and user group information

All the UID and GID, whether generated automatically or in RFC2307 or NIS, are cached. For every positive user login, UID and GID are stored for seven days. For negative user login attempts, where login attempts to the cluster fails, the UID and GID are cached for 2 minutes.

When using external ID mapping server, it is recommended not to change the already created UID or GIDs. If you are required to change the UID or SID, you must plan this activity well, preferably before data exists on IBM Spectrum Scale. After the change, ensure that the cached entries are flushed out.

Caching user and user group information while using LDAP-based authentication

In the LDAP-based user authentication, the **mmuserauth service create** command creates LDAP bind user and bind password in its configuration file (`/etc/pam_ldap.conf`). Using this bind user and password, the 'username' is looked up in the LDAP server with the configured 'login_attribute', which is 'uid' by default. So, '*uid*=<*username*>' is looked up in the LDAP to fetch the DN for this user. If found, the DN and the 'password' are used to perform an LDAP bind operation. The authentication fails if either the '*username*' is not found or the bind operation is failed.

Note: As LDAP is used for UNIX style logins, the user name characters are also matched for the user input versus what is returned from the LDAP server.

For SMB authentication, the privilege of performing the LDAP bind with 'username' and 'password' is not available as the passwords in clear text. The SMB client passes the NT hash for the password. The Samba server matches the NT hash from the client versus the hash fetched from LDAP server. For this, the mmuserauth service create command configures samba registry with the LDAP bind user and password. This user name and password are used to look up the samba-related information from the LDAP server, such as SID, NT hash. If the user name and the hashes are matched, the system grants permission to access the system.

User credentials, group ID mapping, and group membership caching: The user credentials are not cached in the IBM Spectrum Scale system. The user and group ID mapping and group membership cache information are stored in the SSSD component of IBM Spectrum Scale. This cache is for 90 minutes and local to a node and there is no IBM Spectrum Scale native command to purge this cache. However, the sss_cache command (from sssd-common package) might be used to refresh the cache. The sss_cache utility marks only the entries in the respective database as expired. If the LDAP server is online and

available, the expired entries are refreshed immediately. If the LDAP server is not available, the old cached entries that are still marked expired become valid. These values are refreshed once the server is available.

SMB data caching: There is a 7 day cache period for the user/group SID with the Samba component of the IBM Spectrum Scale system. This cache is local to a node and there is no IBM Spectrum Scale native command to purge this cache. However, 'net cache flush' command can be used, per node basis, to purge this. The negative cache entry is for 2 minutes.

Netgroup caching: The netgroup information from the LDAP is also cached for 90 minutes with the SSSD component. The sss_cache command can be used to refresh the cache.

Caching user credentials while using NIS-based authentication for file access

In IBM Spectrum Scale, the NIS server is used only for user and group name, UID/GID, and group membership lookups for NFS access. Also, the netgroups that are defined in NIS are honored for NFS client information in the NFS export configuration. User and group UID and GID information, names, and group membership are cached with the SSSD component of the IBM Spectrum Scale system. This cache is for 90 minutes and local to a node and there is no IBM Spectrum Scale native command to purge this cache.

However, sss_cache command from the sssd-common package might be used to refresh the cache. The sss_cache utility only marks the entries in the respective database as expired. If the NIS server is online and available, the expired entries are refreshed immediately. If the NIS server is not available, the old cached entries that are still marked expired become valid. These values are refreshed once the server is available.

The netgroup information from the NIS is also cached for 90 minutes with the SSSD component. The sss_cache command can be used to refresh the cache.

Caching user and group ID mapping and group membership details while using AD-based authentication

The Samba component of the IBM Spectrum Scale system used the SMB protocol NTLM and Kerberos authentication for user authentication. User and group SID to UID and GID mapping information are cached with the SAMBA component of IBM Spectrum Scale. This cache is maintained for seven days and local to a node and there is no IBM Spectrum Scale native command to purge this cache. However, 'net cache flush' can be used to refresh the cache. The negative cache entry is for 2 minutes.

Group membership cache on IBM Spectrum Scale lies with the Samba component and it is local to each protocol node. For an authenticated user, its group membership cache is valid for the lifetime of that session. The group membership is refreshed on a new authentication request only. If an SMB tree connect is requested on an existing session, the group cache is not refreshed. So, if there are group membership changes made on AD, all the existing session for that use must be disconnected so that there is a new authentication request and cache refresh for the user. If there is no new authentication request that is made by the user, just a simple user and group information look is requested. For example, issue 'id' command on the protocol node, the winbind component of IBM Spectrum Scale searches the database to check whether that user's cached information is there, from any previous authentication request or not. If such an entry is found, that is returned. Otherwise, the winbind fetches the mapping information from AD and caches it for 5 minutes.

Note: If a user was authenticated in the past, its group membership cache information is with the database and valid for a lifetime. Winbind keeps referring this information and will never try to fetch the new information from the AD. To force winbind to return new information, you need to make an authentication request to the node in the same way as connecting from CIFS client.

Authentication for object access

The OpenStack Identity service that is enabled in the system confirms an incoming request by validating a set of credentials that are supplied by the user. The Identity Management consists of both authentication and authorization processes.

In the authentication and authorization process, an object user is identified in the IBM Spectrum Scale system by the attributes such as user ID, password, project ID, role, and domain ID with Keystone API V3. The keystone server that is used for the OpenStack Identity service that is configured on the IBM Spectrum Scale system manages the user authentication requests for object access with the help of either an external or internal authentication server for the user management. An internal or an external keystone server can be configured to manage the identity service. You can use the following authentication methods for object access either by using an internal keystone server shipped IBM Spectrum Scale or by using an external keystone server:

- External authentication. The keystone server interacts with the following external servers for the user management:
  - AD
  - LDAP
- Internal authentication. The keystone server interacts with an internal database to manage users and authentication requests.

The user can select the authentication method based on their requirement. For example, if the enterprise deployment already consists of AD and the users in the AD need to access object data, the customer would use AD as the backend authentication server.

When the authentication method selected is either AD or LDAP, the user management operations such as creating a user, deleting a user are the responsibility of the AD and LDAP administrator. If local authentication is selected for object access, the user management operations must be managed by the Keystone server administrator. The authorization tasks such as defining user roles, creating projects, associating a user with a project are managed by the Keystone administrator. The Keystone server administration can be done either by using Keystone V3 REST API or by using OpenStack python-based client that is called 'openstackclient'.

The following diagram depicts how the authentication requests are handled when IBM Spectrum Scale is with an internal Keystone server and external AD or LDAP authentication server.

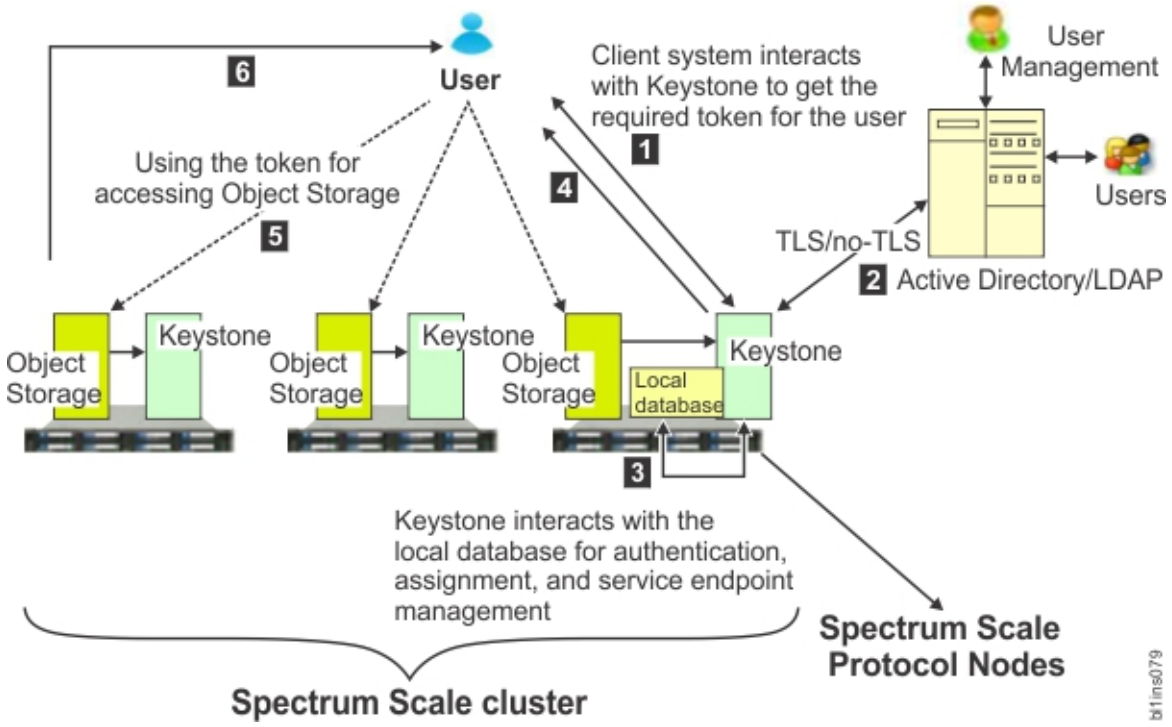


Figure 9. IBM Spectrum Scale integration with internal Keystone server and external AD or LDAP authentication server

The following list provides the authentication process for object access:

1. The user raises the access request to get access to the object data.
2. The keystone server communicates with the authentication server such as AD, LDAP, or a local database. The keystone server interacts with the authentication server for authentication, authorization, and service end-point management.
3. If the user details are valid, the Keystone server interacts with the local database to determine the user roles and issues a token to grant access to the user.
4. The OpenStack Identity service offers token-based authentication for object access. When user credentials are validated, the identity service issues an authentication token, which the user provides in subsequent requests. That is, the access request also includes the token that is granted in step 3. The token is an alphanumeric string of text that is used to access OpenStack APIs and resources.
5. The authenticated user contacts the Object Storage to access the data that is stored in it.
6. The Object Storage grants permission to the user to work on the data based on the associated project ID and user role.

Each object user is part of a project. A project is used to group or isolate resources. Each user needs to be defined with the set of user rights and privileges to perform a specific set of operations on the resources of the project to which it belongs to.

The Identity service also tracks and manages the access to the OpenStack services that are installed on the system. It provides one or more endpoints that can be used to access resources and perform authorized operations. Endpoints are network-accessible addresses (URLs) that can be used to access the service. When the user is authenticated, the keystone server provides a list of services and a token to the user to access the services. For example, if the user is authenticated to access the Object Storage service, the keystone server provides the token to access the service. The Object Storage service then verifies the token and fulfills the request.

To achieve high availability, Object Storage and Keystone are activated on all protocol nodes. The internal database that is required to validate the user is installed on the shared root file system.

Depending upon the configuration, the keystone server needs to interact with AD, LDAP, or the internal database for user authentication. The Keystone internal database is also used for assigning users to projects with a specific role for controlling access to projects. It also holds the definition of OpenStack services and the endpoints for those services.

Each node configured with object storage is configured to interact with the Keystone server.

Deleting authentication and ID mapping

You can choose to delete authentication method and current ID mappings that are configured in the system. Deleting authentication method and ID mappings result in loss of access to data. Before you delete ID mappings, determine how access to data is going to be maintained.

You cannot delete both authentication method and ID mappings together. If you need to delete ID mappings, you need to first delete the authentication method that is configured in the system. If only the authentication settings are deleted, you can return to the existing setting by reconfiguring authentication. There is no recovery procedure for ID mappings. Typically, IDs are deleted in test systems or as part of a service operation.

For more information on how to delete authentication and ID mappings, see *Deleting authentication and ID mapping configuration* in *IBM Spectrum Scale: Administration and Programming Reference*.

Planning for NFS

There are a number of decisions that must be made before deploying NFS in an IBM Spectrum Scale environment. The following is an outline of these decision points.

The IBM Spectrum Scale for NFS architecture is shown in Figure 10 on page 91.

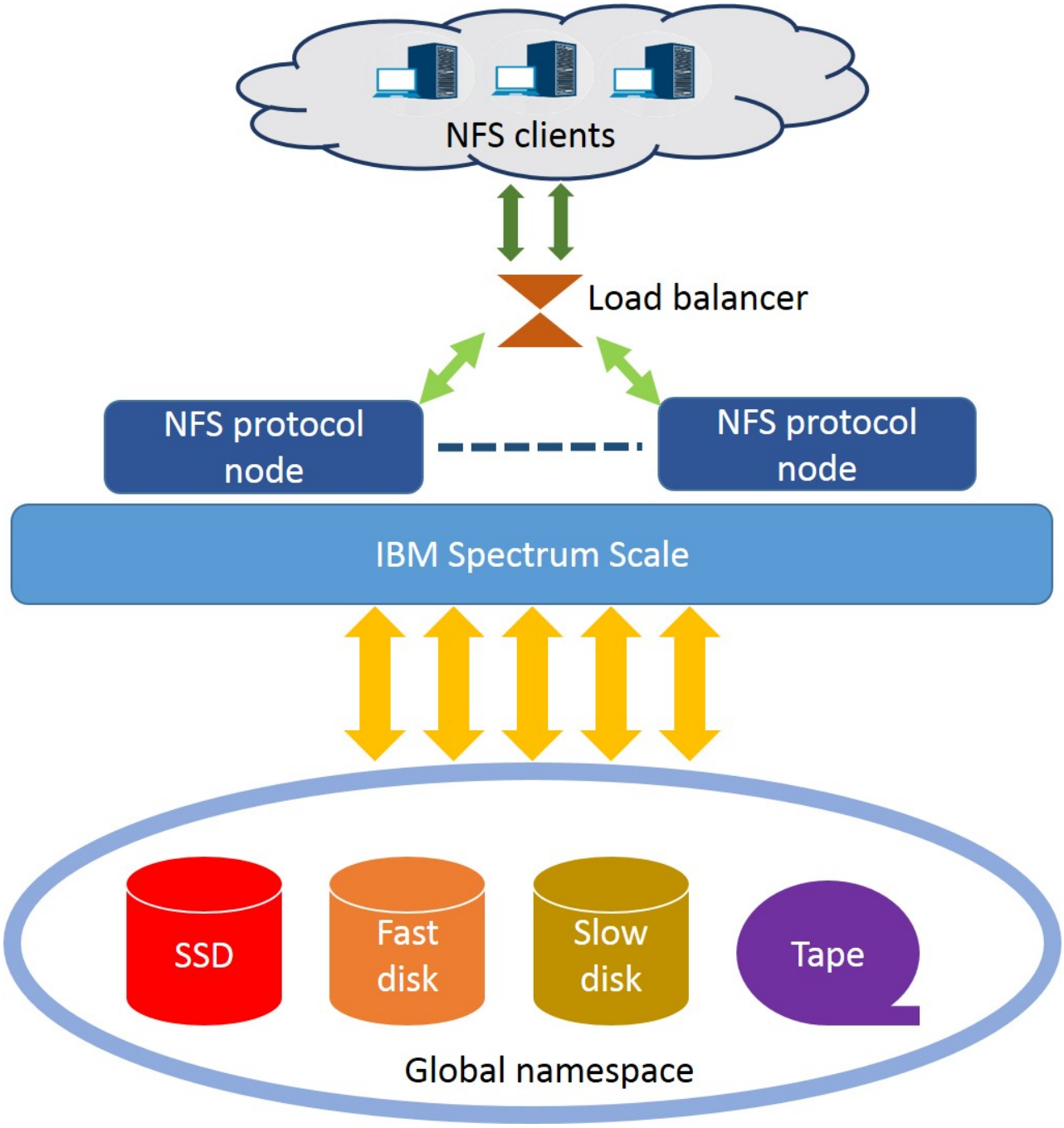


Figure 10. IBM Spectrum Scale for NFS architecture

Each IBM Spectrum Scale protocol node that has NFS enabled, runs the NFS service, other protocol services (SMB and Object, if enabled), and the IBM Spectrum Scale client. NFS users make requests through an NFS client to perform an NFS operation on a mounted NFS file system, for example, to read or to write a file. The request is routed to an NFS protocol node typically by a load balancer or by using DNS round robin. The NFS server on the protocol node implements the request against the backend IBM Spectrum Scale storage file system. The request completion status is then returned to the user through the NFS client.

File system considerations for the NFS protocol

Ensure that all IBM Spectrum Scale file systems used to export data using NFS are mounted with the `syncnfs` option to prevent clients from running into data integrity issues during failover. It is

recommended to use the **mmchfs** command to set the `syncnfs` option as default when mounting the IBM Spectrum Scale file system. For more information, see For more details on these options, see the *mmcjfs command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

For fileset considerations for the NFS protocol, see "Fileset considerations for creating protocol data exports" on page 69.

Considerations for NFS clients

When using NFS clients in an IBM Spectrum Scale environment, the following considerations apply.

- If you mount the same NFS export on one client from two different IBM Spectrum Scale NFS protocol nodes, data corruption might occur.
- IBM Spectrum Scale 4.1.1 and later releases allow concurrent access to the same file data using SMB, NFS, and native POSIX access. For concurrent access to the same data, some limitations apply. For more information, see *Multiprotocol export considerations* in *IBM Spectrum Scale: Administration and Programming Reference*.
- 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.
- It is recommended to use the CES IP address of the IBM Spectrum Scale system to mount the NFS export on an NFS client.

Planning for SMB

This section describes the steps to be taken before using the SMB service in IBM Spectrum Scale.

The SMB support for IBM Spectrum Scale 4.1.1 and later versions allows clients to access the GPFS™ file system using SMB clients. Each SMB protocol node runs the SMB, NFS, and Object services. The SMB service on the protocol node provides file serving to SMB clients. Requests are routed to an SMB protocol node, typically by a load balancer or by using DNS round robin. The SMB server on the protocol node implements these requests by issuing calls to the IBM Spectrum Scale file system.

SMB file serving

IBM Spectrum Scale provides SMB file serving to SMB clients. This topic describes how to manage SMB file serving in IBM Spectrum Scale.

Thousands of clients can concurrently access the SMB exports in an IBM Spectrum Scale system. The system configuration must be able to support a large number of planned SMB connections. For more information on creating protocol data exports, see "Fileset considerations for creating protocol data exports" on page 69.

SMB active connections

Each IBM Spectrum Scale protocol node is capable of handling a large number of active concurrent SMB connections.

The number of concurrent SMB connections that a protocol node can handle is dependent on the following factors:

- Number of processors
- Amount of memory installed in the protocol node
- The IO workload. This depends on the following factors:

  - Number of concurrent SMB connections
  - Lifetime and frequency of SMB connections
  - Overhead due to metadata operations
  - Frequency of operations on each SMB connections
  - Concurrent access to the same files
- The storage configuration and advanced functions that are configured to run while serving high number of SMB connections

SMB fail-over scenarios and upgrade

When you are planning an IBM Spectrum Scale system configuration that has a high number of SMB connections, you must consider the impact that a fail-over can have on the performance of the system.

In the event that an IBM Spectrum Scale protocol node fails, the IP addresses hosted by that protocol node are located to another IBM Spectrum Scale protocol node. SMB clients have to connect to the remaining protocol nodes. These remaining protocol nodes handle all the SMB connections.

Therefore, when you plan an IBM Spectrum Scale system that has a high number of SMB connections, some buffer in terms of number of SMB connections must be factored into the overall system configuration. This will prevent a system overload during a fail-over scenario, thus reducing any adverse effects to the system performance.

A similar consideration applies to SMB upgrades. The SMB upgrade happens in two phases. During the first phase of the upgrade process, the nodes are updated one at a time. The remaining nodes handle the SMB connections of the node being updated. Once all the nodes are updated, the upgrade moves to the second phase. In the second phase, SMB will be shutdown completely. This is done in order to update the SMB code on all the protocol nodes concurrently. This results in a brief outage of the SMB service.

SMB limitations

IBM Spectrum Scale can host a maximum of 1,000 SMB exports. There must be less than 3,000 SMB connections per protocol node and less than 20,000 SMB connections across all protocol nodes.

IBM Spectrum Scale 4.1.1 and later versions allow concurrent access to the same data file through SMB and NFS, and through native POSIX access. However, change notification from protocols other than SMB are not supported for SMB clients. For example, files that are created with NFS are not automatically visible on SMB clients. The SMB clients need to refresh the directory listing by performing a manual refresh in Microsoft Windows Explorer.

Similarly, files that are created using other methods like FTP or restored from a backup, are not automatically visible. To facilitate access to such files from SMB clients, you must set the `gpfs:leases` and `gpfs:sharemodes` options. For more details on these options, see the *mmsmb command* in the *IBM Spectrum Scale: Administration and Programming Reference*.

For more information on SMB client access and limitations, see *Multi-protocol export considerations* in the *IBM Spectrum Scale: Administration and Programming Reference*.

SMB client limitations

Windows: No access for Windows XP or Windows 2003 clients, as SMB protocol version 1 is not supported by IBM Spectrum Scale.

Linux: The Linux kernel SMB client defaults to SMB protocol version 1. However, IBM Spectrum Scale does not support this protocol version. Use the `version` option with the kernel SMB client to set a higher protocol version:

```
mount.cifs //fscc-p8-11/ralph /media/ss –o user=aduser1,pass=Passw0rd,dom=W2K8DOM05,vers=2.0
```

Note: Check with the vendor of your client operating system about the support provided for SMB2 or SMB3 protocols.

SMB share limitations

Consider all the limitations and support restrictions before you create an SMB share. For more information on SMB share limitations, see *SMB share limitations* in the *IBM Spectrum Scale: Administration and Programming Reference*.

SMB node limitations

IBM Spectrum Scale allows a maximum of 16 nodes in a CES cluster.

Planning for object storage deployment

There are a number of decisions that must be made before beginning your object storage deployment in an IBM Spectrum Scale environment. The following is an outline of these decision points.

The IBM Spectrum Scale for object storage architecture is shown in Figure 11.

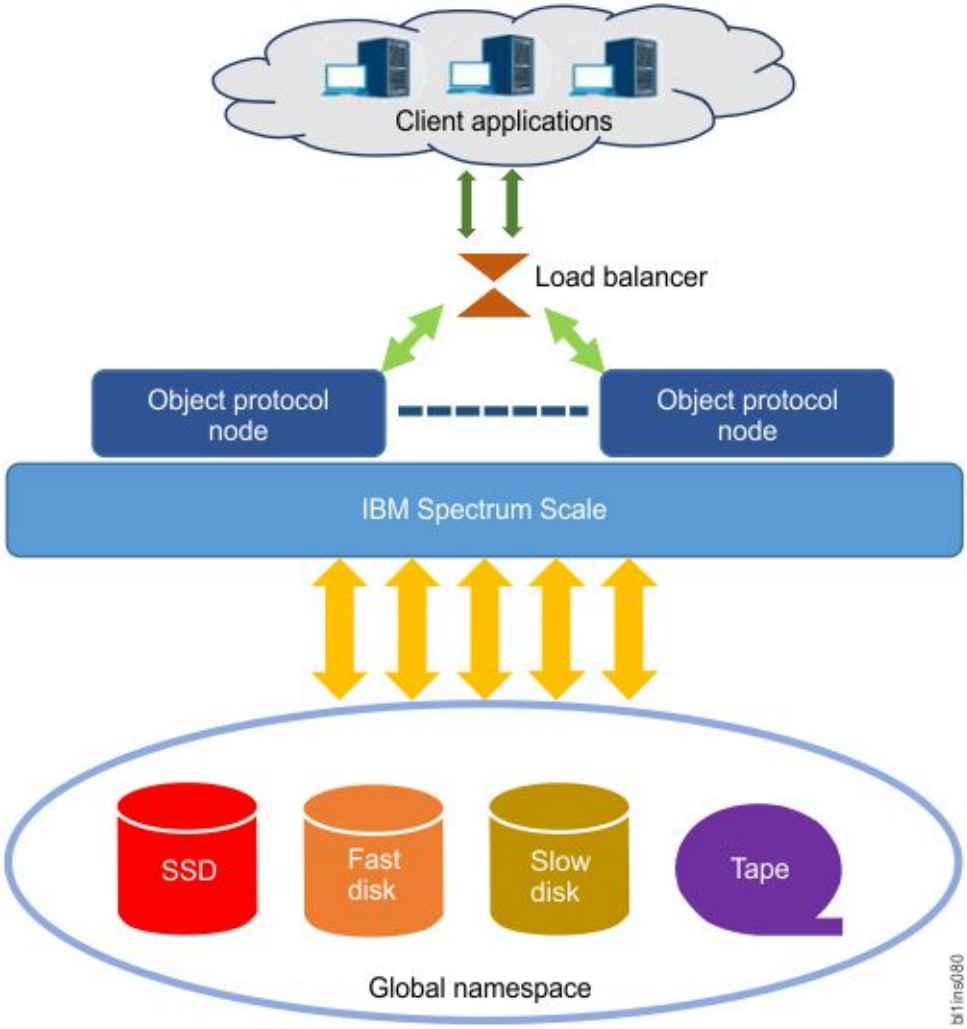


Figure 11. IBM Spectrum Scale for object storage architecture

Each protocol node runs all OpenStack Swift object services, the Keystone identity service, and the IBM Spectrum Scale client. Client applications make requests to perform object operations such as uploading an object, downloading an object, and deleting an object. The request is routed to a protocol node typically by a load balancer or by using DNS round robin. The protocol node implements that request by creating, retrieving, or deleting the object on the backend IBM Spectrum Scale storage. The request completion status is then returned to the client application. Each client request must include an authentication token. Typically, client applications first request a token from the Keystone service, and then provide that token in the subsequent object requests, until the token expires. At that point, a new token can be requested.

For more information on OpenStack Swift, see the OpenStack Swift documentation.

For more information on OpenStack Keystone as it is used with Swift, see the Keystone Auth section in the OpenStack Swift documentation.

Load balancing

Protocol nodes are all active and provide a front-end for the entire object store. Requests can come in directly to the protocol nodes, but it is common to use a load balancer to distribute requests evenly across the nodes.

Any web load balancer product can be used, or DNS Round Robin can be used. In either case, there is an IP address and a corresponding **cluster host name** that is used to identify the object storage cluster. This is the address clients communicate with, and the load balancer (or DNS) routes client requests to a particular protocol node.

Cluster host name

The cluster host name is required during the installation process.

It is the **endpoint** parameter in the **spectrumscale config object** command and it is the **ces_hostname** parameter in the **mmobj swift base** command. Additionally, the cluster host name might be used in your load balancer or DNS round robin configuration. It is the fully qualified name that clients send their object requests to.

Authentication method

IBM Spectrum Scale for object storage supports a flexible array of configuration options for authentication.

If you already have Keystone deployed in your environment, you can configure IBM Spectrum Scale for object storage to use the existing or external Keystone. If you do configure Keystone as part your IBM Spectrum Scale cluster, you can manage the user information locally, or you can integrate it with a Microsoft Active Directory or LDAP system.

Backup and disaster recovery strategy

For information on backup and disaster recovery strategy for your object storage, see the *Protocols cluster disaster recovery* section in the *IBM Spectrum Scale: Advanced Administration Guide*.

SELinux considerations

To simplify the configuration of the IBM Spectrum Scale for object storage environment, the installation process updates SELinux settings so that the object services and the database software running on the protocol nodes can interact with the required file system and system resources.

The `openstack-selinux` package is installed automatically when the `spectrum-scale-object` RPM is installed. This configures the object services for SELinux.

If the installer detects that SELinux is enabled, it does the following steps:

1. Ensures that the Postgres database can access the Keystone database directory on the CES shared root file system:

```
semanage fcontext -a -t postgresql_db_t "<keystone db directory>(/.*)?"
semanage fcontext -a -t postgresql_log_t "<keystone db directory>/log(/.*)?"
restorecon -R "<keystone db directory>"
```

2. Ensures that object processes can access the object data fileset:

```
semanage fcontext -a -t swift_data_t "<object fileset directory>(/.*)?"
restorecon -R <object fileset directory>/*
```

Attention: If you want to use SELinux in the Enforcing mode, you must take that decision before proceeding with the deployment. Changing the SELinux mode after the deployment is not supported.

SELinux packages required for IBM Spectrum Scale for object storage

IBM Spectrum Scale for object storage requires the following SELinux packages to be installed otherwise the installation using the **spectrumscale** installation toolkit fails because the dependency cannot be met:

- `selinux-policy-base` at 3.13.1-23 or higher
- `selinux-policy-targeted` at 3.12.1-153 or higher

Eventual consistency model

An eventual consistency model is used when uploading or deleting object data.

According to the CAP theorem, in distributed storage systems, a system can guarantee two of the following three claims:

1. Consistency
2. Availability
3. Partition tolerance

With Swift and IBM Spectrum Scale for object storage, the availability and partition tolerance claims are chosen. Therefore, in some cases, it is possible to get an inconsistent listing of object store contents. The most common case is consistency between container listing databases and objects on disk. In the case of a normal object upload, the object data is committed to the storage and the container listing database is updated. Under heavy load, it is possible that the container listing database update does not complete immediately. In that case, the update is made asynchronously by the object-updater service. In the time between the original object commit and when the object-updater executes, you will not see the new object in the container listing, even though the object is safely committed to storage. A similar situation can occur with object deletes.

Planning for unified file and object access

Unified file and object access allows use cases where you can access data using object as well as file interfaces. Before using unified file and object access, you need to plan for a number of aspects.

Planning for identity management modes for unified file and object access:

As you plan for unified file and object access, it is important to understand the identity management modes for unified file and object access.

Based on the identity management mode that you plan to use, you need to plan for the authentication mechanism to be configured with file and object. In an existing IBM Spectrum Scale setup, where an authentication mechanism is already set up, a suitable identity management mode needs to be chosen for unified file and object access.

For more information, see *Identity management modes for unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

Authentication planning for unified file and object access:

Planning for a suitable authentication mechanism for a unified file and object access setup requires an understanding of the Identity management modes for unified file and object access and the authentication setups supported with these modes.

For more information, see *Authentication in unified file and object access* and *Identity management modes for unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

ibmobjectizer service schedule planning:

The unified file and object access feature allows accessing data ingested through file to be accessed from object. This access is enabled by a process called objectization. Objectization is done by the **ibmobjectizer** service that runs periodically and converts the file data located under unified file and object access enabled filesets to make it available for object access.

It is important to understand the amount and frequency of file data that is expected to be ingested and objectized because the objectization service needs to be scheduled accordingly by the administrator.

For information about the **ibmobjectizer** service, see *The objectizer process*, *Setting up the objectizer service schedule*, and *Configuration files for IBM Spectrum Scale for object storage* in *IBM Spectrum Scale: Administration and Programming Reference*.

Prerequisites for unified file and object access:

To enable unified file and object access, you must enable the `file-access` object capability.

For more information see "Object capabilities" on page 34.

For unified file and object access, the authentication prerequisites depend on the identity management mode for unified file and object access that you plan to use. It is important to decide the mode before configuring unified file and object access, although it is possible to move from one mode to another. For more information on changing identity management modes for unified file and object access, see *Configuring authentication and setting identity management modes for unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

For more information, see *Authentication in unified file and object access* and *Identity management modes for unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

Unified file and object access is deployed as a storage policy for object storage. Therefore, it is important to understand the concept of a storage policy for object storage and how to associate containers with storage policies. For more information, see "Storage policies for object storage" on page 31.

Planning for multi-region object deployment

Use the following information to plan your multi-region object deployment.

Enabling multi-region enables the primary rings to be multi-region and all data to be stored in all regions. Storage policies can be used to limit objects to certain regions.

For information on enabling multi-region object deployment, see "Enabling multi-region object deployment initially" on page 133.

For information on adding a new region to a multi-region object deployment environment, see *Adding a region in a multi-region object deployment* in *IBM Spectrum Scale: Administration and Programming Reference*.

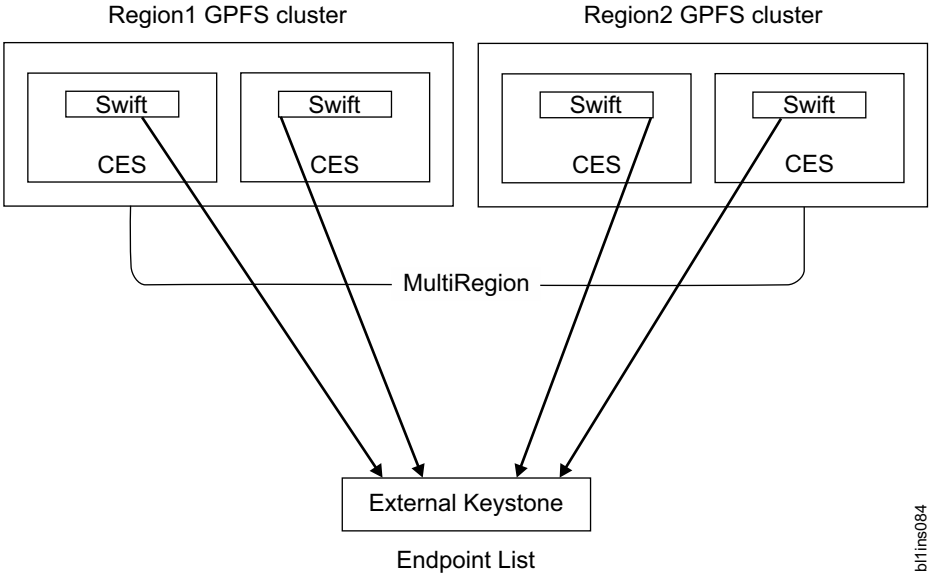
Authentication considerations for multi-region object deployment:

In a multi-region object deployment environment, all regions must use the same Keystone service.

The Keystone service can be local Keystone installed with the object deployment or it can be an independent service. Subsequent clusters which join the environment must specify an external Keystone server during installation.

The following two methods can be used for object authentication configuration with a multi-region setup:

- By using the external keystone



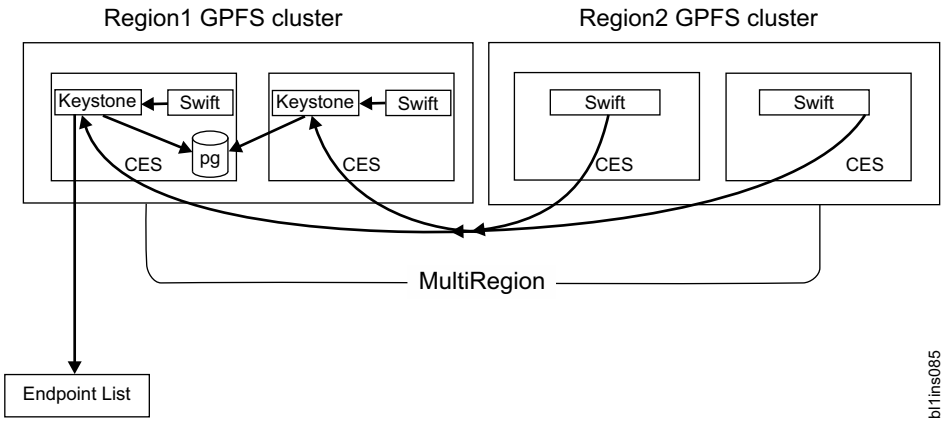

```
+------+-----------+--------------+--------------+---------+-----------+-----------------------------------------+
| ID   | Region    | Service Name | Service Type | Enabled | Interface | URL                                     |
+------+-----------+--------------+--------------+---------+-----------+-----------------------------------------+
| e310 | RegionOne | swift        | object-store | True    | public    | http://region1:8080/v1/AUTH_%(tenant_id)s |
| 1679 | RegionOne | swift        | object-store | True    | internal  | http://region1:8080/v1/AUTH_%(tenant_id)s |
| c458 | RegionOne | swift        | object-store | True    | admin     | http://region1:8080                     |
| 8a01 | Region2   | swift        | object-store | True    | public    | http://region2:8080/v1/AUTH_%(tenant_id)s |
| b821 | Region2   | swift        | object-store | True    | internal  | http://region2:8080/v1/AUTH_%(tenant_id)s |
| 5188 | Region2   | swift        | object-store | True    | admin     | http://region2:8080                     |
+------+-----------+--------------+--------------+---------+-----------+-----------------------------------------+
```

Note: The external keystone and HA must be managed and configured by the customers.

1. On all the participant clusters of the multi-region setup, configure the external keystone with the **spectrumscale auth object external** command.
2. Set the **keystone_url** and **configure_remote_keystone** properties.
3. For manual installation, use the **mmobj swift base** command with the **--remote-keystone-url** and **--configure-remote-keystone** arguments.

Note: The installer can automatically create these endpoints if the option to configure the remote keystone is used during installation and **–configure-remote-keystone** is specified.

- By using the keystone installed on one of the spectrum scale clusters




```
+------+-----------+-------------+-------------+---------+-----------+-----------------------------------------+
| ID   | Region    | Service Name | Service Type | Enabled | Interface | URL                                     |
+------+-----------+-------------+-------------+---------+-----------+-----------------------------------------+
| e310 | RegionOne | swift        | object-store | True    | public    | http://region1:8080/v1/AUTH_%(tenant_id)s |
| 1679 | RegionOne | swift        | object-store | True    | internal  | http://region1:8080/v1/AUTH_%(tenant_id)s |
| c458 | RegionOne | swift        | object-store | True    | admin     | http://region1:8080                     |
| 8a01 | Region2   | swift        | object-store | True    | public    | http://region2:8080/v1/AUTH_%(tenant_id)s |
| b821 | Region2   | swift        | object-store | True    | internal  | http://region2:8080/v1/AUTH_%(tenant_id)s |
| 5188 | Region2   | swift        | object-store | True    | admin     | http://region2:8080                     |
+------+-----------+-------------+-------------+---------+-----------+-----------------------------------------+
```

Note: If the region1 cluster stops functioning, the complete multi-region setup will be unusable because the keystone service is not available.

1. On the first cluster of multi-region setup, configure local Keystone with the **spectrumscale auth object local|ad|ldap** command by using the spectrumscale installation toolkit.
2. For manual installation, use the **mmobj swift base** command with the **--local-keystone** arguments for configuring with keystone with local authentication type.
3. For configuring the object authentication with ad | ldap, use **mmuserauth service create|delete** command after **mmobj swift base with -- local-keystone**.
4. On the second and third clusters of the multi-region setup, configure the external keystone with the **spectrumscale auth object external** command.
5. Set the **keystone_url** and **configure_remote_keystone** properties.
6. For manual installation, use the **mmobj swift base** command with the **--remote-keystone-url** and **--configure-remote-keystone** arguments.

Note: The installer can automatically create these endpoints if the option to configure remote keystone is used during installation if **--configure-remote-keystone** is specified.

Data protection considerations for multi-region object deployment: Each cluster must maintain appropriate backups. A multi-region cluster can be more resilient to failures because if a cluster loses multi-region objects or a single cluster fails, the lost objects are restored through the normal Swift replication behavior.

Since all regions use the same Keystone service, ensure the Keystone data is backed up appropriately.

Network considerations for multi-region object deployment:

To communicate with nodes in other regions, the Swift services connect to the network addresses as defined in the ring files, which are the CES IP addresses. This means that every node must be able to connect to all CES IPs in all regions.

Ensure that the network routing is properly set up to enable the connections. Ensure that the necessary firewall configuration is done to allow connection to the object, account, and container servers (typically

ports 6200, 6201, and 6202) on all nodes in all regions. Swift uses **rsync** to replicate data between regions, so the **rsync** port 873 also needs to be opened between the regions.

Monitoring and callbacks in an multi-region object deployment setup: Within a region, the monitoring of multi-region ring files and distribution to nodes is dependent upon the existing monitoring framework and the work done to enable storage policies in the monitoring layer. For the distribution of ring and configuration changes to other regions, the administrator needs to perform those steps manually using the **mmobj multiregion** command.

Performance considerations for multi-region object deployment: Before objects are replicated from one cluster to another, there is a delay because of the replicator run time and the network load.

Each region's access to the Keystone server affects the performance of authentication capabilities within its region.

Chapter 3. Installing IBM Spectrum Scale and deploying protocols on Linux nodes

This section explains how to install IBM Spectrum Scale and deploy protocols on Linux nodes.

Steps to establishing and starting your GPFS cluster

There are several steps you must perform to establish and start your GPFS cluster. This topic provides the information you need for performing those steps manually.

The **spectrumscale** installation toolkit, available for RHEL7, automates many of the steps listed below.

You can install GPFS and deploy protocols either manually or by using the **spectrumscale** installation toolkit. This topic provides the information you need for establishing and starting your GPFS cluster manually. If you have already installed GPFS with the spectrumscale toolkit, these steps have already been completed.

Follow these steps to establish your GPFS cluster:

1. Review supported hardware, software, and limits by reviewing the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) for the latest recommendations on establishing a GPFS cluster.
2. Install the GPFS licensed program on your system:
   - For existing systems, see Chapter 6, "Migration, coexistence and compatibility," on page 171.
   - For new systems:
     - For your Linux nodes, see "Installing GPFS on Linux nodes" on page 102.
     - For your AIX nodes, see Chapter 4, "Installing IBM Spectrum Scale on AIX nodes," on page 153.
     - For your Windows nodes, see Chapter 5, "Installing IBM Spectrum Scale on Windows nodes," on page 157.
3. Decide which nodes in your system will be quorum nodes (see "Quorum" on page 41).
4. Create your GPFS cluster by issuing the **mmcrcluster** command. See "GPFS cluster creation considerations" on page 47.
5. Use the **mmchlicense** command to assign an appropriate GPFS license to each of the nodes in the cluster. See "IBM Spectrum Scale license designation" on page 80 for more information.

If you used the **spectrumscale** installation toolkit to install GPFS, steps 2 through 5, and optionally, step 6 below have already been completed.

After your GPFS cluster has been established:

1. Ensure you have configured and tuned your system according to the values suggested in the *Configuring and tuning your system for GPFS* topic in *IBM Spectrum Scale: Administration and Programming Reference*.
2. Start GPFS by issuing the **mmstartup** command. For more information, see **mmstartup command** in *IBM Spectrum Scale: Administration and Programming Reference*.
3. Create new disks for use in your file systems by issuing the **mmcrnsd** command. See "Network Shared Disk (NSD) creation considerations" on page 52.
4. Create new file systems by issuing the **mmcrfs** command. See "File system creation considerations" on page 56.
5. Mount your file systems.

© Copyright IBM Corporation © IBM 2014, 2016

6. As an optional step, you can also create a temporary directory (**/tmp/mmfs**) to collect problem determination data. The **/tmp/mmfs** directory can be a symbolic link to another location if more space can be found there. If you decide to do so, the temporary directory should *not* be placed in a GPFS file system, as it might not be available if GPFS fails.

   If a problem should occur, GPFS might write 200 MB or more of problem determination data into **/tmp/mmfs**. These files must be manually removed when any problem determination is complete. This should be done promptly so that a **NOSPACE** condition is not encountered if another failure occurs. An alternate path can be specified by issuing the **mmchconfig dataStructureDump** command.

Installing GPFS on Linux nodes

There are three steps to installing GPFS on Linux nodes: Preparing the environment; installing the GPFS software; and building the GPFS portability layer. The information in this topic points you to the detailed steps.

Before installing GPFS, you should review "Planning for GPFS" on page 39 and the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Installing GPFS without ensuring that the prerequisites listed in "Hardware requirements" on page 39 and "Software requirements" on page 40 are satisfied can lead to undesired results.

The installation process includes:

1. "Preparing the environment on Linux nodes"
2. "Preparing to install the GPFS software on Linux nodes" on page 103
3. "Building the GPFS portability layer on Linux nodes" on page 149
4. "For Linux on z Systems: Changing the kernel settings" on page 151

Preparing the environment on Linux nodes

Before proceeding with installation, prepare your environment by following the suggestions in the following sections.

Add the GPFS bin directory to your shell PATH

Ensure that the PATH environment variable for the root user on each node includes **/usr/lpp/mmfs/bin**. (This is not required for the operation of GPFS, but it can simplify administration.)

Other suggestions for cluster administration

GPFS commands operate on all nodes required to perform tasks. When you are administering a cluster, it may be useful to have a more general form of running commands on all of the nodes. One suggested way to do this is to use an OS utility like **dsh** or **pdsh** that can execute commands on all nodes in the cluster. For example, you can use **dsh** to check the kernel version of each node in your cluster:

```
# dsh uname -opr
Node01: 2.6.18-128.1.14.e15 x86_64 GNU/Linux
Node02: 2.6.18-128.1.14.e15 x86_64 GNU/Linux
```

Once you have **dsh** set up, you can use it to install GPFS on a large cluster. For details about setting up **dsh** or a similar utility, review the documentation for the utility.

Verify that prerequisite software is installed

Before installing GPFS, it is necessary to verify that you have the correct levels of the prerequisite software installed on each node in the cluster. If the correct level of prerequisite software is *not* installed, see the appropriate installation manual before proceeding with your GPFS installation.

For the most up-to-date list of prerequisite software, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

The FAQ contains the latest information about the following:

- Supported Linux distributions and kernel levels
- Recommended or required RPM levels
- Software recommendations
- Configuration information

Before proceeding, see also "GPFS and network communication" on page 16.

Preparing to install the GPFS software on Linux nodes

Follow the steps in this topic in the specified order to install the GPFS software.

This procedure installs GPFS on one node at a time:

1. "Accepting the electronic license agreement on Linux nodes"
2. "Extracting the GPFS software on Linux nodes"
3. "Extracting GPFS patches (update SLES or Red Hat Enterprise Linux RPMs or Debian Linux packages)" on page 105
4. "Installing the GPFS man pages on Linux nodes" on page 106
5. "Deciding whether to install GPFS and deploy protocols manually or with the spectrumscale installation toolkit" on page 107

   In this step, you can choose to install GPFS and protocols either manually or using the **spectrumscale** installation toolkit.
6. "Verifying the GPFS installation on SLES and Red Hat Enterprise Linux nodes" on page 138

   or

   "Verifying the GPFS installation on Debian Linux nodes" on page 137

Accepting the electronic license agreement on Linux nodes

The GPFS software license agreement is shipped with the GPFS software and is viewable electronically. When you extract the GPFS software, you are asked whether or not you accept the license.

The electronic license agreement must be accepted before software installation can continue. Read the software agreement carefully before accepting the license. See "Extracting the GPFS software on Linux nodes."

Extracting the GPFS software on Linux nodes

The GPFS software is delivered in a self-extracting archive.

The self-extracting image contains the following.

- The IBM Spectrum Scale product installation SLES and Red Hat Enterprise Linux RPMs and Debian Linux packages

  Note: Debian Linux packages are not available for z Systems.
- The License Acceptance Process (LAP) Tool

The LAP Tool is invoked for acceptance of the IBM Spectrum Scale license agreements. The license agreements must be accepted to obtain access to the IBM Spectrum Scale product installation images.

- A version of the Java™ Runtime Environment (JRE) necessary to run the LAP Tool

1. Copy the self-extracting product image from the DVD to a local directory, specifying the correct version of the product for your hardware platform and Linux distribution; for example:

   ```
   cp /media/dvd/Spectrum_Scale_Standard-4.2.0.0-x86_64-Linux-install/
   tmp/Linux/Spectrum_Scale_Standard-4.2.0-0_x86_64-Linux-install
   ```

2. Verify that the self-extracting program has executable permissions, for example:

   ```
   # ls -l /tmp/Spectrum_Scale_Standard-4.2.0-0_x86_64-Linux-install
   ```

   The system displays information similar to the following:

   ```
   -rwxr-xr-x 1 root root 673341648 Nov 10 13:53 /tmp/Spectrum_Scale_Standard-4.2.0-0_x86_64-Linux-install
   ```

   If it is not executable, you can make it executable by issuing the **chmod +x** command.

   Note: To get a list of all the rpms in the self-extracting package, you can issue the following command (specifying the **--manifest** option):

   ```
   Spectrum_Scale_Standard-4.2.0-0_x86_64-Linux-install --manifest
   ```

   The system displays output like the following:

   ```
   manifestsw,rpm,gpfs.gskit-8.0.50-47.x86_64.rpm  Tue 10 Nov 2015
       09:08:39 AM MST md5sum:ddd28945883579c26d947d32cc9b818asw,rpm,gpfs.docs-4.2.0-0.noarch.rpm
       Tue 10 Nov 2015 09:08:17 AM MST
       md5sum:e181f9d78f6ae6d761c4217c97387723sw,rpm,gpfs.msg.en_US-4.2.0-0.noarch.rpm  Tue 10
       Nov 2015 09:08:19 AM MST
       md5sum:a0d042ccea75915e6069661fe9c444desw,rpm,gpfs.gui-4.2.0-0.sles12.x86_64.rpm  Mon 02
       Nov 2015 08:28:52 AM MST
       md5sum:27d2e4b8a921e1eb56600b95dbbf7428sw,rpm,gpfs.base-4.2.0-0.x86_64.rpm  Tue 10 Nov
       2015 09:01:11 AM MST
       md5sum:b961ffdc79031a9c242d04550fccdc97sw,rpm,gpfs.callhome-jre-8.0-1.0.x86_64.rpm  Tue 27
       Oct 2015 01:41:46 PM MST
       md5sum:31a6df8ca66ee44156f5f6150b21b62csw,rpm,gpfs.ext-4.2.0-0.x86_64.rpm  Tue 10 Nov 2015
       09:08:30 AM MST
       md5sum:f8da698505f87dd17072a5d1cafef04bsw,rpm,gpfs.callhome-4.2-1.002.noarch.rpm  Tue 27
       Oct 2015 01:41:46 PM MST
       md5sum:969445b18ad53c36bb1979de489211b0sw,rpm,gpfs.gss.pmcollector-4.2.0-0.SLES12.x86_64.rpm
       Tue 20 Oct 2015 08:13:39 AM MST
       md5sum:6d8635539ed975e697e07c6516dc2ea7sw,rpm,gpfs.callhome-ecc-client-4.2-1.0.noarch.rpm
       Tue 27 Oct 2015 01:41:46 PM MST
       md5sum:a72f961deac0826ce4119b6e6a2adc35sw,rpm,gpfs.gpl-4.2.0-0.noarch.rpm  Tue 10 Nov 2015
       09:13:22 AM MST
       md5sum:367ea797bbbef7942a52e057ef3526desw,rpm,gpfs.gss.pmcollector-4.2.0-0.el7.x86_64.rpm
       Tue 20 Oct 2015 08:13:40 AM MST
       md5sum:38ae895404201a447a75b4ee034beae9sw,rpm,gpfs.gss.pmsensors-4.2.0-0.SLES12.x86_64.rpm
       Tue 20 Oct 2015 08:13:40 AM MST
       md5sum:e9c2b46086c4d91085e6afad1e978826sw,rpm,gpfs.gss.pmsensors-4.2.0-0.el7.x86_64.rpm
       Tue 20 Oct 2015 08:13:41 AM MST
       md5sum:6a5e64ee109630c74568f6cb3bacf9ccsw,rpm,gpfs.gui-4.2.0-0.el7.x86_64.rpm  Mon 02 Nov
       2015 08:14:33 AM MST md5sum:a48e7a686a8a44ac1052e7a8d2797d44...
       ...
   ```

   (The remaining rpms in the package are displayed.)

3. Invoke the self-extracting image that you copied from the DVD by using a command such as **Spectrum-Scale-Standard-4.2.0.0-x86_64-Linux-install** and accept the license agreement:
   a. By default, the LAP Tool, JRE, and GPFS installation images are extracted to the target directory **/usr/lpp/mmfs/4.2.0.0**.
   b. The license agreement files on the media can be viewed in text-only mode or graphics mode:
      - Text-only is the default mode. When run the output explains how to accept the agreement:

```
<...Last few lines of output...>
Press Enter to continue viewing the license agreement, or
enter "1" to accept the agreement, "2" to decline it, "3"
to print it, "4" to read non-IBM terms, or "99" to go back
to the previous screen.
```

- To view the files in graphics mode, invoke **Spectrum-Scale-Standard-4.2.0.0-x86_64-Linux-install**. Using the graphics-mode installation requires a window manager to be configured.

c. You can use the **--silent** option to accept the license agreement automatically.

d. Use the **--help** option to obtain usage information from the self-extracting archive.

Upon license agreement acceptance, the IBM Spectrum Scale product installation images are placed in the extraction target directory (**/usr/lpp/mmfs/4.2.0.0**). This directory contains the GPFS SLES and Red Hat Enterprise Linux RPMs that are applicable for the Spectrum Scale edition that you installed (Express, Standard, or Advanced).

Note: For this release, the IBM Global Security Kit (GSKit) version for RPMs and Debian Linux packages must be at least `8.0.50.47` or higher.

In this directory there is a **license** subdirectory that contains license agreements in multiple languages. To view which languages are provided, issue the following command:

```
ls /usr/lpp/mmfs/4.2.0.0/license
```

The system displays information similar to the following:

```
Chinese_TW.txt French.txt  Japanese.txt        notices.txt   Slovenian.txt
Chinese.txt    German.txt  Korean.txt          Polish.txt    Spanish.txt
Czech.txt      Greek.txt   Lithuanian.txt      Portuguese.txt Status.dat
English.txt    Indonesian.txt   Italian.txt non_ibm_license.txt Russian.txt   Turkish.txt
```

The license agreement remains available in the extraction target directory under the **license** subdirectory for future access. The license files are written using operating system-specific code pages. This enables you to view the license in English and in the local language configured on your machine. The other languages are not guaranteed to be viewable.

Extracting GPFS patches (update SLES or Red Hat Enterprise Linux RPMs or Debian Linux packages)

Typically when you install a GPFS system there are patches available. It is recommended that you always look for the latest patches when installing or updating a GPFS node.

Note: For information about restrictions pertaining to Debian Linux, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

GPFS patches (update SLES and Red Hat Enterprise Linux RPMs and Debian Linux packages) are available from the IBM Support Portal: Downloads for General Parallel File System (www.ibm.com/support/entry/portal/Downloads/Software/Cluster_software/General_Parallel_File_System).

GPFS patches (update SLES and Red Hat Enterprise Linux RPMs and Debian Linux packages) are available from Fix Central http://www-933.ibm.com/support/fixcentral/i

The update SLES and Red Hat Enterprise Linux RPMs and Debian Linux packages are distributed as self-extracting base software.

If you are applying a patch during the initial installation of GPFS on a node, you only need to build the portability layer once after the base and update SLES or Red Hat Enterprise Linux RPMs or Debian Linux packages are installed.

Installing the GPFS man pages on Linux nodes

In order to use the GPFS man pages, the **gpfs.docs** RPM must be installed.

Once you have installed the **gpfs.docs** RPM, the GPFS man pages are located at **/usr/share/man/**.

You do not need to install the **gpfs.docs** RPM on all nodes if man pages are not desired (for example, if local file system space on the node is minimal).

If you are using RHEL7, the **spectrumscale** installation toolkit automatically installs the associated man pages when it runs **spectrumscale install** to create a new GPFS cluster.

Overview of the spectrumscale installation toolkit

The **spectrumscale** installation toolkit automates the steps required to install GPFS, deploy protocols, and install updates and patches.

When using the **spectrumscale** installation toolkit, you provide environmental information and the toolkit will install, configure, and deploy the optimal configuration, dynamically creating a cluster definition file.

This install toolkit enables you to do the following:

- Install and configure GPFS.
- Add GPFS nodes to an existing cluster.
- Deploy and configure SMB, NFS, Object, and performance monitoring tools on top of GPFS.
- Configure authentication services for protocols.
- Upgrade GPFS and all protocols and install patches.

Installation and configuration are driven through commands.

In the self-extracting package, the **spectrumscale** installation toolkit is in this location:
location_extracted_to/4.2.0.0/installer

Using the **spectrumscale** installation toolkit is driven from the **spectrumscale** executable in the installer directory, and this can optionally be added to the path.

Note: The toolkit installs the Chef configuration management tool, a Python-based tool wrapped around Opscode Chef technologies, enabling configuration management and deployment at scale. For more information, see Apache license information (www.apache.org/licenses/LICENSE-2.0).

The **spectrumscale** installation toolkit operation consists of four phases:

1. User input via **spectrumscale** commands
   a. All user input is recorded into a clusterdefinition.txt file in /usr/lpp/mmfs/4.2.0.0/installer/configuration
   b. Please review the clusterdefinition.txt file to make sure that it accurately reflects your cluster configuration
   c. As you input your cluster configuration, remember that you can have the toolkit act on parts of the cluster by simply not telling it about nodes that may have incompatible OS levels, architectures, etc.
2. A **spectrumscale** install phase
   a. Install will act upon all nodes input into the clusterdefinition.txt file
   b. GPFS and perfmon rpms will be installed
   c. GPFS portability layer will be created
   d. GPFS will be started

e. A cluster will be created
f. Server and client licenses will be applied
g. GUI nodes may be created and the GUI may be started upon these nodes
h. NTP, perfmon, GPFS ephemeral ports, and cluster profile may be configured
i. NSDs may be created - Note that file systems are not created during install

3. A spectrumscale deploy phase
   a. Deploy will act upon all nodes input into the `clusterdefinition.txt` file
   b. File systems will be configured - Note that it is possible to only configure file systems during the deploy phase if you do not want protocols
   c. SMB, NFS, and Object protocol rpms will be copied to all protocol nodes
   d. SMB, NFS, and Object services may be started
   e. Authentication may be configured
   f. Server and client licenses will be applied
   g. GUI nodes may be created and the GUI may be started upon these nodes
   h. NTP, perfmon, GPFS ephemeral ports, and cluster profile may be configured
   i. NSDs may be created - Note that file systems are not created during install
4. A spectrumscale upgrade phase
   a. Upgrade will act upon all nodes input into the `clusterdefinition.txt` file
   b. All installed/deployed components will be upgraded
   c. Upgrades are sequential with multiple passes
   d. Pass 1 of all nodes will upgrade GPFS sequentially
   e. Pass 2 of all nodes will upgrade NFS sequentially
   f. Pass 3 of all nodes will upgrade SMB sequentially
   g. Pass 4 of all nodes will upgrade Object sequentially
   h. The Spectrum Scale GUI may be installed and started upon upgrade

Deciding whether to install GPFS and deploy protocols manually or with the spectrumscale installation toolkit

Starting with IBM Spectrum Scale 4.1.1 on Red Hat Enterprise Linux 7, you can install GPFS and deploy protocols either manually or using the **`spectrumscale`** installation toolkit.

Why would I want to use the `spectrumscale` installation toolkit?

While planning your installation, consider the advantages provided by the **`spectrumscale`** installation toolkit. The installation toolkit:

1. Simplifies GPFS cluster creation
2. Automatically creates NSD and file system stanza files
3. Creates new NSDs, file systems, and adds new nodes
4. Has a single package manager for all components of a release
5. Deploys Object, SMB, NFS by automatically pulling in all pre-requisites
6. Configures file and Object authentication during deployment
7. Consistently sets and syncs time on all nodes
8. Deploys the Spectrum Scale GUI
9. Configures Performance Monitoring consistently across all nodes
10. Simplifies upgrade with a single command to upgrade all components on all nodes.

Note: The **`spectrumscale`** installation toolkit supports installation, deployment, and upgrading of both IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition.

The toolkit does not install, deploy, or upgrade packages specific to the IBM Spectrum Scale Advanced Edition. They need to be done manually after the **spectrumscale** installation toolkit is done. The edition level will be Standard until the manual installation is complete and IBM Spectrum Scale is restarted.

Before deciding which method you want to use, review the following topics:

- "Assumptions/prerequisites and packaging" on page 109
- "Understanding the **spectrumscale** installation toolkit options" on page 112
- "Using the **spectrumscale** installation toolkit to perform installation tasks: Explanations and examples" on page 114
- "Manually installing the GPFS software packages on Linux nodes" on page 135

Installing IBM Spectrum Scale by using the graphical user interface (GUI)

You can use the GUI to install and configure IBM Spectrum Scale system. The installation GUI is used only for installing the system and a separate management GUI needs to used for configuring and managing the system.

The scope of the installation GUI is to install the IBM Spectrum Scale software on cluster nodes, create an IBM Spectrum Scale cluster, and configure NTP. Other initial tasks like creating file systems, creating NSDs, setting up NFS and SMB protocols, object storage, or authentication of protocol users are out of scope of the installation GUI. You need to perform those tasks either through CLI or management GUI.

The prerequisites that are applicable for installing the IBM Spectrum Scale system through CLI is applicable for installation through the GUI as well. For more information on the prerequisites for installation, see "Installation prerequisites" on page 109.

The installation GUI requires port 9080 to be designated for http and port 9443 for https. That is, the installer GUI URL of the http and https connections are `http://<install server IP or host name>:9080` and `https://<install server IP or host name>:9443` respectively.

Note: To start the graphical installation, the IBM Spectrum Scale package must be extracted.

By selecting different parameters of the **spectrumscale installgui** command, you can start, stop, and view the status of the installation GUI.

```
spectrumscale installgui [-h] {start,stop,status} ...
```

Example:

To start the installation GUI, issue the following command:

```
./spectrumscale installgui start
```

To view the status of the processes that are running on the installation GUI, issue the following command:

```
./spectrumscale installgui status
```

The installation process through the GUI automatically stops when you exit the installation GUI. To stop installation process through the CLI, issue the following command:

```
./spectrumscale installgui stop
```

Note: The password to access the IBM Spectrum Scale installer GUI is `Passw0rd`.

Limitations of the installation GUI

The following are the limitations of the installation GUI:

- The scope of the installation GUI is to install the IBM Spectrum Scale software on cluster nodes, create an IBM Spectrum Scale cluster, and configure NTP. Other initial tasks like creating file systems, creating NSDs, setting up NFS and SMB protocols, object storage, or authentication of protocol users are out of scope of the installation GUI. You need to perform those tasks either through CLI or management GUI.
- It supports only a fresh installation of an IBM Spectrum Scale cluster. That is, it does not support to update the system from a previous release to the latest release.
- It does not support adding nodes to and removing nodes from the existing IBM Spectrum Scale cluster
- You cannot uninstall the existing IBM Spectrum Scale cluster through the GUI.

Assumptions/prerequisites and packaging

Review the installation prerequisites and packaging overview before proceeding with your installation.

This information is provided in the following topics:

- "Installation prerequisites"
- "Packaging overview" on page 112

Note: See "Extracting the GPFS software on Linux nodes" on page 103.

Installation prerequisites

There are several prerequisites for installing IBM Spectrum Scale.

These prerequisites are the following:

Packages

The **spectrumscale** installation toolkit requires one of the following packages:

- python-2.6 with argparse
- python-2.7

Operating system requirements

The **spectrumscale** installation toolkit supports Red Hat Enterprise Linux 7.0 and 7.1 platforms on x86_64 and ppc64 architectures.

Red Hat Enterprise Linux 7 must be loaded on all nodes that will be designated as protocol nodes. The installation toolkit and instructions within this document will not work with earlier versions of Red Hat Enterprise Linux nor with other vendor operating systems at this time.

Cleanup required if you previously used the Object Redpaper™ configuration process

If you have already configured the system with Openstack software to use GPFS (following instructions from the Object Red paper) be sure to follow the Object cleanup steps in "Cleanup procedures required if reinstalling with the spectrumscale installation toolkit" on page 194 before you start any installation activity.

GPFS 4.2.0

If you want to run protocols in a mixed-version cluster, the protocol nodes and all of the quorum nodes must run 4.2.0; other nodes can run an older version.

Note: For Object protocol, the GPFS software is not required on the node that hosts the Keystone identity service if it is deployed on a separate node from the GPFS protocol nodes.

Additional GPFS requirements

- If you want to run protocols, CCR must be available.
- **mmchconfig release=LATEST** must be run.
- A GPFS cluster and file system are required. If this infrastructure does not already exist, you must install the GPFS software, create a GPFS cluster and create a file system. You can use the **spectrumscale** installation toolkit to do this. See "Using the **spectrumscale** installation toolkit to perform installation tasks: Explanations and examples" on page 114.

- The GPFS file system must be mounted on all GPFS protocol nodes.
- It is strongly recommended to configure the file system to only support NFSv4 ACLs. You can use the **spectrumscale** installation toolkit to do this if you use it to install GPFS. See "Using the **spectrumscale** installation toolkit to perform installation tasks: Explanations and examples" on page 114.

  Alternatively, you can use the **-k nfs4** parameter for **mmcrfs**. NFSv4 ACLs are a requirement for ACL usage with the SMB and NFS protocols. For more information, see **mmcrfs** examples in the *IBM Spectrum Scale: Administration and Programming Reference*.
- Quotas are not enabled by default in GPFS file systems but are recommended for use with SMB and NFS protocols. For more information about quota management, see *Enabling and disabling GPFS quota management* in *IBM Spectrum Scale: Administration and Programming Reference*.
- For Object, the installation toolkit creates an independent fileset in the GPFS file system that you name.

Creation of a file system or fileset or path for shared root

The installation toolkit uses a shared root storage area to install the protocols on each node. This storage is also used by NFS and object protocols to maintain system data associated with the cluster integration we provide. This storage can be a subdirectory in an existing file system or it can be a file system on its own. Once this option is set, changing it will require a restart of GPFS.

You can use the **spectrumscale** installation toolkit to set up this shared root storage area if you use the toolkit for GPFS installation and file system creation. See "Using the **spectrumscale** installation toolkit to perform installation tasks: Explanations and examples" on page 114.

However, if you want to set up shared root before launching the installation toolkit, the following steps can be used:

1. Create a file system or fileset for shared root. Size must be at least 4 GB
2. **mmchconfig cesSharedRoot=***path_to_the_filesystem/fileset_created_in_step_1*

SSH and network setup

This is a general GPFS prerequisite, not specific to protocols other than protocol-specific ports that need to be open on the firewall.

The node used for initiating installation of SMB, NFS, and/or Object protocols via the installation toolkit must be able to communicate via an internal or external network with all protocol nodes to be installed. All nodes also require SSH keys to be set up so that the installation toolkit can run remote commands without any prompts. Examples of common prompts are a prompt asking for a remote node's password and/or a prompt asking a Y/N question. No prompts should exist when using SSH among any cluster nodes to and from each other, and to and from the node designated for installation.

While reading through the following firewall port prerequisites, refer to this example for opening the Red Hat 7 firewall :

```
$ firewall-cmd --permanent --add-port=lowerlimit-upperlimit/tcp
$ firewall-cmd --permanent --add-port=lowerlimit-upperlimit/tcp
$ firewall-cmd --permanent --add-port=lowerlimit-upperlimit/udp
$ firewall-cmd --add-port=lowerlimit-upperlimit/tcp
$ firewall-cmd –add-port=lowerlimit-upperlimit/udp
```

After any firewall changes are done, on any node that these changes were done, the firewall service needs to be restarted (so you might want to do this once after applying all the relevant firewall changes).

```
$ systemctl restart firewalld.service
```

The installation toolkit uses TCP port 8889 for communicating with the Chef server. This port must be reachable from all protocol nodes. The installation toolkit does not change any SELinux settings; the user must change those settings, if required.

The SMB protocol uses external TCP port 445 and must be reachable by clients on all protocol nodes. The internal TCP port 4379 is used for internal communication between the protocol nodes and therefore must be reachable from all protocol nodes. If SMB is enabled and these ports are not open, the installation toolkit will fail during precheck. The failure will indicate to make these ports accessible and retry.

The Object protocol uses external TCP ports 5000 for the keystone server and 8080 for the proxy server. If Object is enabled, both must be reachable from all protocol nodes. Internal TCP ports used by object include 6200 (object server), 6201 (container server), 6202 (account server), 35357 (keystone), 11211 (memcached), 5431 (Object Postgres instance). All internal ports needed by the Object protocol must be reachable within the cluster from all protocol nodes.

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.

GPFS cluster operation requires port 1191 be opened and reachable among all nodes within the internal GPFS cluster network. In addition, an ephemeral port range must be opened as well.

```
$ mmchconfig tscCmdPortRange=lowerlimit-upperlimit
```

Follow up by opening these ports on each node as well.

See the topics about GPFS port usage and firewall recommendations in the *IBM Spectrum Scale: Advanced Administration Guide* in IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html).

Repository Setup

This is a general GPFS prerequisite, not specific to protocols.

The **spectrumscale** installation toolkit contains all necessary code for installation; however, there may be base RHEL7 rpms required as prerequisites. In order to satisfy any prerequisites, it may be necessary for your RHEL7 nodes to have access to a DVD repository (or an external repository accessible by network). Repositories containing IBM Spectrum Scale dependencies include the following x86 examples:

```
rhel-x86_64-server-7 Red Hat Enterprise Linux Server
```

For more information about Yum repositories, see Configuring Yum and Yum Repositories (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/sec-Configuring_Yum_and_Yum_Repositories.html).

NTP Setup

It is recommended that Network Time Protocol (NTP) be configured on all nodes in your system to ensure that the clocks of all of the nodes are synchronized. Clocks that are not synchronized will cause debugging issues and authentication problems with the protocols.

If you are using the install toolkit for IBM Spectrum Scale installation, then no prior NTP configuration is required other than the NTP package is installed and the NTP daemon (ntpd) is not running on all nodes. Use the **config ntp** commands listed in Understanding the spectrumscale installation toolkit options, Table 1 to set NTP servers and then NTP will automatically be configured at install time.
However, if you are manually installing IBM Spectrum Scale or would prefer to manually configure NTP, see the following example of enabling NTP on a node.

```
# yum install -y ntp
# ntpdate <NTP_server_IP>
# systemctl enable ntpd
# systemctl start ntpd
# timedatectl list-timezones
# timedatectl set-timezone
# systemctl enable ntpd
```

Performance Monitoring tool
This is a general GPFS prerequisite, not specific to protocols.

You need to ensure that the boost-regex package is installed in the system before installation of the Performance Monitoring tool.

Collection of core data
This is a general GPFS prerequisite, not specific to protocols.

See the topic about changing configurations to enable **spectrumscale** to collect core dump data in "Understanding the **spectrumscale** installation toolkit options."

Packaging overview

The IBM Spectrum Scale self-extracting package consists of several components.

These components are the following:

Installation toolkit and license
The components necessary to begin the protocol installation.

GPFS GPFS version 4.2.0, included in this self-extracting package. This version must be installed and configured on all protocol nodes prior to initiating the installation toolkit.

NFS (Ganesha)
The rpms for Ganesha, the open-source, user-space implementation of the NFS protocol. They can be installed, enabled, and configured using the installation toolkit.

SMB (Samba)
The rpms for Samba, the open-source implementation of the SMB protocol. They can be installed and enabled using the installation toolkit.

Object (Swift and Keystone)
The Swift and Keystone rpms necessary for enablement of Object services. They can be installed, enabled, and configured using the installation toolkit.

Performance monitoring tool
The rpms necessary for performance monitoring by GPFS (including protocols). They can be installed and enabled using the installation toolkit.

Understanding the `spectrumscale` installation toolkit options

Use the following information to understand how to work with the **spectrumscale** installation toolkit, and toolkit options and limitations.

When you use the **spectrumscale** installation toolkit to *install GPFS*, you do so in two stages:

1. Using a series of **spectrumscale** commands to add node and NSD specifications and configuration properties to the cluster definition file.
2. Using the **spectrumscale install** command to install GPFS on the nodes specified in the cluster definition file and to apply the configuration options to the cluster.

When you use the **spectrumscale** installation toolkit to *deploy protocols*, you do so in two similar stages:

1. Using a series of **spectrumscale** commands to specify environment details, to identify which protocols are to be enabled, and to define protocol-specific properties in the cluster definition file.
2. Using the **spectrumscale deploy** command to deploy protocols as specified in the cluster definition file.

If you already have an existing GPFS cluster, with GPFS started and at least one file system for the CES shared file system, you can just define protocol nodes in the cluster definition and then deploy protocols.

The topic "Using the **spectrumscale** installation toolkit to perform installation tasks: Explanations and examples" on page 114 shows how to use these command options (listed in Table 8 on page 113) to

perform the installation and deployment. After you review the examples, you can tailor them according to your own needs and then proceed to implement them.

Table 8. **spectrumscale** *command options for installing GPFS and deploying protocols*

spectrumscale command option	Purpose
node add	Adds node specifications to the cluster definition file.
nsd add	Adds NSD specifications to the cluster definition file.
nsd clear	Clears all NSDs.
nsd delete	Removes a single NSD.
nsd list	Lists all NSDs currently in the configuration.
nsd modify	Modifies an NSD.
filesystem list	Lists all file systems that currently have NSDs assigned to them.
filesystem modify	Changes the block size and mount point of a file system.
auth file	Configures file authentication on protocols in the cluster definition file.
auth object	Specifies Object authentication on protocols in the cluster definition file.
config gpfs	Adds GPFS-specific properties to the cluster definition file.
install	Installs GPFS on the configured nodes, creates a cluster, and creates NSDs.
config protocols	Provides details about the GPFS environment to be used during protocol deployment.
config perfmon	Configures performance monitoring settings
config object	Defines object-specific properties to be applied during deployment.
deploy	Creates file systems and deploys protocols on your configured nodes.
config ntp	Configures NTP settings
upgrade	Upgrades the various components of the installation.

For additional details about these command options, see the **spectrumscale** command description in the *IBM Spectrum Scale: Administration and Programming Reference*.

Limitations of the spectrumscale installation toolkit

Before using the **spectrumscale** installation toolkit to install GPFS and deploy protocols, review the following limitations.

- The toolkit does currently not support disabling protocols, uninstalling protocols, or uninstalling GPFS on an existing GPFS cluster. This must be done manually:
  - For more detailed information about removing exports, see *Managing protocol data exports* in *IBM Spectrum Scale: Administration and Programming Reference*.
  - For more detailed information about disabling protocol services, see *Managing protocol services* in *IBM Spectrum Scale: Administration and Programming Reference*.
  - For information about uninstalling GPFS, see Chapter 8, “Steps to permanently uninstall GPFS and/or Protocols,” on page 193.
- The toolkit currently supports only key-based SSH authentication. If host-based SSH authentication is configured, either set up key-based SSH authentication temporarily for use with the toolkit, or follow the manual steps in “Manually installing the GPFS software packages on Linux nodes” on page 135.

- The toolkit will not configure any exports on SMB or NFS. For information about configuring Cluster Export Services and creating exports, see *Configuring Cluster Export Services* and *Managing protocol data exports* in *IBM Spectrum Scale: Administration and Programming Reference*.
- The toolkit does not support multiple clusters being defined in the cluster definition.
- The toolkit does not currently install or manage packages specific to the IBM Spectrum Scale Advanced Edition such as the Crypto package. To use these features after using the installation toolkit you will need to install or upgrade them manually by following the steps in "Manually installing the GPFS software packages on Linux nodes" on page 135.
- The toolkit does not support authentication reconfiguration. Also, the toolkit does not use the authentication section of the cluster definition file during upgrade. If you want to change the authentication method, see *Modifying authentication method* in *IBM Spectrum Scale: Administration and Programming Reference*.

Configuration changes required to enable `spectrumscale` installation toolkit to collect core dump data

In order to collect core dumps for debugging programs in provided packages, the following system configuration changes need to be made on all protocol nodes in the cluster:

1. Install the `abrt-cli` rpm if not already installed. For example, run **rpm -qa | grep abrt-cli** to check if already installed, or **yum install abrt-cli** to install the rpm.
2. Set **OpenGPGCheck=no** in the **/etc/abrt/abrt-action-save-package-data.conf** file.
3. Set **MaxCrashReportsSize = 0** in the **/etc/abrt/abrt.conf** file.
4. Start (or restart) the abort daemon (for example, run **systemctl start abrtd** to start the abort daemon after a new install, or **systemctl restart abrtd** if the daemon was already running and the values in steps 2 and 3 were changed).

For additional details, refer to documentation about ABRT-specific configuration (https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/sect-abrt-configuration-abrt.html).

Additional setup steps applicable for NFS

A core dump might not be generated for code areas where the CES NFS process has changed credentials. To avoid this, do the following steps:

1. Insert the following entry into the `/etc/sysctl.conf` file:
   ```
   fs.suid_dumpable = 2
   ```
2. Issue the following command to refresh with the new configuration:
   ```
   sysctl -p
   ```
3. Verify that `/proc/sys/fs/suid_dumpable` is correctly set:
   ```
   cat /proc/sys/fs/suid_dumpable
   ```
   Note: The system displays the following output if it is correctly set:
   ```
   2
   ```

If you are reinstalling with the spectrumscale installation toolkit

If you are reinstalling with the **`spectrumscale`** installation toolkit, you must perform some cleanup procedures first. See "Cleanup procedures required if reinstalling with the spectrumscale installation toolkit" on page 194.

Using the spectrumscale installation toolkit to perform installation tasks: Explanations and examples

This topic explains and provides examples of **`spectrumscale`** installation toolkit options and tasks.

After you have determined how you want your own system to be configured, and after you have reviewed the **spectrumscale** command description in *IBM Spectrum Scale: Administration and Programming Reference*, you can tailor these examples to do the following:

- Set up your install node.

- Add node and NSD specifications, file system information, and GPFS configuration properties to your cluster definition file, and then install GPFS and configure your cluster according to the information in that file.
- Specify environment details, identify which protocols are to be enabled, and define protocol-specific properties in the cluster definition file, and then deploy protocols on your system.

Notes:

1. The **spectrumscale** installation toolkit requires one of the following packages:
   - python-2.6 with argparse
   - python-2.7
2. The **spectrumscale** installation toolkit supports RHEL 7.0 and 7.1 platforms on x86_64 and ppc64 architectures.

Setting up the install node

The first step in using the **spectrumscale** installation toolkit for installing IBM Spectrum Scale and deploying protocols is to configure your install node.

A good candidate for your install node is the GPFS `admin` node, because a prerequisite of this node is that it must be able to communicate with all other nodes without the need to provide a password. Also, SSH connections between the `admin` node and all other nodes must be set up to suppress prompting. Therefore, no prompts should exist when using ssh among any cluster nodes to and from each other, and to and from the server.

First, find the install toolkit by changing directories to where it was extracted (the default 4.2.0.0 extraction path is shown below. This path may vary, depending on the code level).

```
cd/ usr/lpp/mmfs/4.2.0.0/installer
```

To configure the install node, issue the following command:

```
./spectrumscale setup -s ServerIP -i SSHIdentity
```

The **-s** argument identifies the IP that nodes will use to retrieve their configuration. This IP will be one associated with a device on the install node. (This is automatically validated during the setup phase.)

Optionally, you can specify a private SSH key to be used to communicate with nodes in the cluster definition file, using the **-i** argument.

As part of the setup process, the appropriate Chef packages will be deployed on the install node.

Defining configuration options for the spectrumscale installation toolkit

Use these instructions to set up the cluster definition file prior to installing GPFS and deploying protocols.

Adding node definitions to the cluster definition file

GPFS node definitions are added to the cluster definition file through the **spectrumscale node add** command.

1. If the install toolkit is being used from a location outside of any of the nodes to be installed, a GPFS Admin Node is required. The Admin Node will be used to run GPFS cluster-wide commands.

   To specify a GPFS Admin Node in the cluster definition file, use the **-a** argument:

   ```
   ./spectrumscale node add gpfsnode1 -a
   ```

   If no GPFS Admin Node is specified in the cluster definition file, the node the install toolkit is running on is automatically designated as the Admin Node. If GUI nodes are to be installed, each GUI node must also be marked as an Admin node.

   The role of an Admin node with regards to the installation toolkit is to serve as the coordinator of the installation, deployment, and upgrade. This node will also act as a central repository for all Spectrum Scale rpms. For larger clusters, it is important to have an Admin node with plenty of network bandwidth to all other nodes in the cluster.

2. To add GPFS Client nodes to the cluster definition file, provide no arguments:

   ```
   ./spectrumscale node add gpfsnode1
   ```

3. To add GPFS manager nodes to the cluster definition file, use the **-m** argument:

   ```
   ./spectrumscale node add gpfsnode2 -m
   ```

   If no manager nodes are added to the cluster definition, the install toolkit will automatically designate manager nodes using the following algorithm:

   a. First, all protocol nodes in the cluster definition will be designated as manager nodes.
   b. If there are no protocol nodes, all NSD nodes in the cluster definition will be designated as manager nodes.
   c. If there are no NSD nodes, all nodes in the cluster definition will be designated as manager nodes.

4. GPFS quorum nodes are added to the cluster definition using the **-q** argument.

   ```
   ./spectrumscale node add gpfsnode3 -q
   ```

   If no quorum nodes are added to the cluster definition, the install toolkit will automatically designate quorum nodes using the following algorithm:

   a. If the number of nodes in the cluster definition is less than 4, all nodes will be designated as quorum nodes.
   b. If the number of nodes in the cluster definition is between 4 and 9 inclusive, 3 nodes will be designated as quorum nodes.
   c. If the number of nodes in the cluster definition is between 10 and 18 inclusive, 5 nodes will be designated as quorum nodes.
   d. If the number of nodes in the cluster definition is greater than 18, 7 nodes will be designated as quorum nodes.

   This algorithm will preferentially select NSD nodes as quorum nodes. If the number of NSD nodes is less than the number of quorum nodes to be designated then any other nodes will be selected until the number of quorum nodes is satisfied.

5. GPFS NSD servers are added to the cluster definition using the **-n** argument.

   ```
   ./spectrumscale node add gpfsnode4 -n
   ```

6. GPFS Graphical User Interface servers are added to the cluster definition using the **-g** argument.

   ```
   ./spectrumscale node add gpfsnode3 -g
   ```

   A GUI server must also be an admin node. Use the -a flag:

   ```
   ./spectrumscale node add gpfsnode3 -a
   ```

   If no nodes have been specified as management GUI servers, then the GUI will not be installed. It is recommended to have at least 2 management GUI interface servers and a maximum of 3 for redundancy.

7. To display a list of all nodes in the cluster definition file, use the **spectrumscale node list** command. For example:

   ```
   $ ./spectrumscale node list
   [ INFO  ] List of nodes in current configuration:
   [ INFO  ] [Installer Node]
   [ INFO  ] 9.168.100.1
   [ INFO  ]
   [ INFO  ] [Cluster Name]
   [ INFO  ] gpfscluster01
   [ INFO  ]
   [ INFO  ] GPFS Node               Admin  Quorum  Manager  NSD Server  Protocol Management Interface
   [ INFO  ] gpfsnode1                 X       X                                          x
   [ INFO  ] gpfsnode2                                                                    x
   [ INFO  ] gpfsnode3                         X       X         X
   [ INFO  ] gpfsnode4                         X       X         X
   ```

Adding and configuring NSD server nodes in the cluster definition file

Note: A CES shared root file system is required for protocols deployment with IBM Spectrum Scale.

1. To configure NSDs, you must first have added your NSD server nodes to the configuration:

```
./spectrumscale node add -n nsdserver1
./spectrumscale node add -n nsdserver2
```

2. Once NSD server nodes are in the configuration, you can add NSDs to the configuration.

   ```
   ./spectrumscale nsd add /dev/sdb -p nsdserver1 -s nsdserver2
   ```

   The install toolkit supports standalone NSDs which connect to a single NSD server or shared NSDs which connect to both a primary and secondary NSD server.

   When adding a standalone NSD, skip the secondary NSD server parameter.

   When adding a shared NSD, it is important to know the device name on the node which is to become the primary NSD server. It is not necessary to know the device name on the secondary NSD server because the device will be looked up using its UUID.

   Note: Although it is not necessary to know the device name on the secondary NSD server, it may be helpful to create a consistent mapping of device names if you are using multipath. For more information, see "NSD disk discovery" on page 24.

   Here is an example of adding a shared NSD to the configuration by specifying the device name on the primary server along with the primary and secondary servers.

3. The name of the NSD will be automatically generated based on the NSD server names. This can be changed after the NSD has been added by using the modify command and supplying a new name to the **-n** flag; the new name must be unique:

   ```
   ./spectrumscale nsd modify nsdserver1_nsdserver2_1 -n new_name
   ```

4. It is possible to view all NSDs currently in the configuration using the list command:

   ```
   ./spectrumscale nsd list
   ```

5. To remove a single NSD from the configuration, supply the name of the NSD to the delete command:

   ```
   ./spectrumscale nsd delete nsdserver1_nsdserver2_1
   ```

6. To clear all NSDs and start from scratch, use the clear command:

   ```
   ./spectrumscale nsd clear
   ```

7. Where multiple devices are connected to the same pair of NSD servers, they can be added in bulk either by providing a list of all devices, or by using wild cards:

   ```
   ./spectrumscale nsd add -p nsdserver1 -s nsdserver2 /dev/dm-1 /dev/dm-2 /dev/dm-3
   or
   ./spectrumscale nsd add -p nsdserver1 -s nsdserver2 "/dev/dm-*"
   ```

   A connection will be made to the primary server to expand any wild cards and check that all devices are present. When using wild cards, it is important to ensure that they are properly escaped, as otherwise they may be expanded locally by your shell. If any devices listed cannot be located on the primary server, a warning will be displayed, but the command will continue to add all other NSDs.

8. When adding NSDs, it is good practice to have them distributed such that each pair of NSD servers is equally loaded. This is usually done by using one server as a primary for half of the NSDs, and the other server as primary for the remainder. To simplify this process, it is possible to add all NSDs at once, then later use the balance command to switch the primary and secondary servers on some of the NSDs, as required. A connection will be made to the original secondary server to look up device names on that node automatically. To automatically balance a pair of NSD servers, you must specify one of the nodes in that pair:

   ```
   $ ./spectrumscale nsd add "/dev/dm-*" -p serverA -s serverB
   [ INFO  ] Connecting to serverA to check devices and expand wildcards.
   [ INFO  ] Adding NSD serverA_serverB_1 on serverA using device /dev/dm-0.
   [ INFO  ] Adding NSD serverA_serverB_2 on serverA using device /dev/dm-1.
   $ ./spectrumscale nsd list
   [ INFO  ] Name                          FS      Size(GB) Usage   FG Pool    Device    Servers
   [ INFO  ] serverA_serverB_1 Default 13       Default 1  Default /dev/dm-0 serverA,serverB
   [ INFO  ] serverA_serverB_2 Default 1        Default 1  Default /dev/dm-1 serverA,serverB
   $ ./spectrumscale nsd balance --node serverB
   $ ./spectrumscale nsd list
   ```

```
[ INFO  ] Name                          FS      Size(GB) Usage   FG Pool    Device    Servers
[ INFO  ] serverA_serverB_1 Default 13       Default 1  Default /dev/dm-0 serverB,serverA
[ INFO  ] serverA_serverB_2 Default 1        Default 1  Default /dev/dm-1 serverA,serverB
```

9. Ordinarily a connection will be made to the primary NSD server when adding an NSD. This is done to check device names and so that details such as the disk size can be determined, but is not vital. If it is not feasible to have a connection to the nodes while adding NSDs to the configuration, these connections can be disabled using the **--no-check** flag. Extra care is needed to manually check the configuration when using this flag.

   ```
   ./spectrumscale nsd add /dev/sda -p nsdserver1 -s nsdserver2 --no-check
   ```

10. You can set the failure group, file system, pool, and usage of an NSD in two ways:
    - using the add command to set them for multiple new NSDs at once
    - using the modify command to modify one NSD at a time

```
./spectrumscale nsd add "/dev/dm-*" -p nsdserver1 -s nsdserver2 -po pool1 -u dataOnly -fg 1 -fs filesystem_1
./spectrumscale nsd modify nsdserver1_nsdserver2_1 -u metadataOnly -fs filesystem_1
```

Creating file systems

File systems are defined implicitly by the NSD configuration and will only be created if there are NSDs assigned to them.

Note: Be aware that if you do not configure file systems, the installation toolkit will automatically create a file system named `scale0` mounted at `/ibm/scale0/`.

Note: A CES shared root file system is required for protocols deployment with IBM Spectrum Scale.

1. To create a file system, use the **nsd add** or **nsd modify** command to set the file system property of the NSD:

   ```
   $ ./spectrumscale nsd add "/dev/dm-*" -p server1 -s server2 -fs filesystem_1
   [ INFO  ] The installer will create the new file system filesystem_1 if it does not already exist.
   $ ./spectrumscale nsd modify server1_server2_1 -fs filesystem_2
   [ INFO  ] The installer will create the new file system filesystem_2 if it does not already exist.
   ```

2. To list all file systems that currently have NSDs assigned to them, use the **list** command. This will also display file system properties including the block size and mount point:

   ```
   $ ./spectrumscale filesystem list
   [ INFO  ] Name         BlockSize   Mountpoint            NSDs Assigned
   [ INFO  ] filesystem_1 Default     /ibm/filesystem_1     3
   [ INFO  ] filesystem_2 Default     /ibm/filesystem_2     1
   ```

3. To alter the block size and mount point from their default values, use the modify command:

   ```
   ./spectrumscale filesystem modify filesystem_1 -b 1M -m /gpfs/gpfs0
   ```

   NSDs will be created when the **spectrumscale install** command is issued.

   The file system will be created when the **spectrumscale deploy** command is issued.

   It is not possible to directly rename or delete a file system; this is instead done by reassigning the NSDs to a different file system using the **nsd modify** command.

At this point, the `clusterdefinition.txt` file will now have:

- nodes and node types defined
- NSDs optionally defined
- File systems optionally defined

Go to the next step, Installing GPFS, to proceed with the GPFS installation.

Installing GPFS

After setting up your install node, you can use the **spectrumscale** installation toolkit to define nodes, NSDs, and file systems in the cluster definition file, and install GPFS according to the information in that file.

Steps for installing GPFS

After defining nodes in the cluster definition file (see "Defining configuration options for the spectrumscale installation toolkit" on page 115), you can set additional GPFS configuration information in that file. You use the **spectrumscale config gpfs** command to do this.

1. To specify a cluster name in the cluster definition, use the **-c** argument:

   ```
   ./spectrumscale config gpfs -c gpfscluster01.my.domain.name.com
   ```

   If no cluster name is specified, the GPFS Admin Node name is used as the cluster name. If the user-provided name contains periods, it is assumed to be a fully qualified domain name. If the cluster name is not a fully qualified domain name, the cluster name domain name will be inherited from the Admin Node domain name.

2. To specify a profile to be set on cluster creation in the cluster definition use the **-p** argument:

   ```
   ./spectrumscale config gpfs -p randomio
   ```

   The valid values for -p option are default (for **gpfsProtocolDefaults** profile) and random I/O (for **gpfsProtocolRandomIO** profile). The profiles are based on workload type : sequential I/O (**gpfsProtocolDefaults**) or random I/O (**gpfsProtocolRandomIO**). The defined profiles above can be used to provide initial default tunables/settings for a cluster. If additional tunable changes are required, see **mmchconfig** command and the **mmcrcluster** command in *IBM Spectrum Scale: Administration and Programming Reference*.

   If no profile is specified in the cluster definition, the **gpfsProtocolDefaults** profile will be automatically set on cluster creation.

3. To specify the remote shell binary to be used by GPFS, use the **-r** argument:

   ```
   ./spectrumscale config gpfs -r /usr/bin/ssh
   ```

   If no remote shell is specified in the cluster definition, `/usr/bin/ssh` will be used as default.

4. To specify the remote file copy binary to be used by GPFS use the **-rc** argument:

   ```
   ./spectrumscale config gpfs -rc /usr/bin/scp
   ```

   If no remote file copy binary is specified in the cluster definition, `/usr/bin/scp` will be used a default.

5. To specify an ephemeral port range to be set on all GPFS nodes use the **-e** argument:

   ```
   spectrumscale config gpfs -e 70000-80000
   ```

   For information about the ephemeral port range, see *GPFS port usage* in *IBM Spectrum Scale: Advanced Administration Guide*.

   If no port range is specified in the cluster definition, 60000-61000 will be used as default.

6. To view the current GPFS configuration settings, issue the following command:

   ```
   $ ./spectrumscale config gpfs --list
   [ INFO  ] No changes made. Current settings are as follows:
   [ INFO  ] GPFS cluster name is gpfscluster01
   [ INFO  ] GPFS profile is default
   [ INFO  ] Remote shell command is /usr/bin/ssh
   [ INFO  ] Remote file copy command is /usr/bin/scp
   [ INFO  ] GPFS Daemon communication port range is 60000-61000
   ```

To perform environment checks prior to running the install, use **spectrumscale install** with the **-pr** argument:

```
./spectrumscale install -pr
```

This is not required, however, as **install** with no arguments will also run this.

Understanding what the install toolkit does during a spectrumscale install

- If the **spectrumscale** installation toolkit is being used to install GPFS on all nodes, create a new GPFS cluster, and create NSDs, it will automatically perform the steps outlined below.
- If the **spectrumscale** installation toolkit is being used to add nodes to an existing GPFS cluster and/or create new NSDs, it will automatically perform the steps outlined below.

- If all nodes in the cluster definition file are in a cluster, then the **spectrumscale** installation toolkit will automatically perform the steps outlined below.

To add nodes to an existing GPFS cluster, at least one node in the cluster definition must belong to the cluster where the nodes not in a cluster are to be added. The GPFS cluster name in the cluster definition must also exactly match the cluster name outputted by **mmlscluster**.

Once the **spectrumscale install** command has been issued, the toolkit will follow these flows:

To install GPFS on all nodes, create a new GPFS cluster, and create NSDs

- Run pre-install environment checks
- Install the GPFS packages on all nodes
- Build the GPFS portability layer on all nodes
- Install and configure performance monitoring tools
- Create a GPFS cluster
- Configure licenses
- Set ephemeral port range
- Create NSDs (if any are defined in the cluster definition)
- Run post-install environment checks

To add nodes to an existing GPFS cluster and create any new NSDs

- Run pre-install environment checks
- Install the GPFS packages on nodes to be added to the cluster
- Install and configure performance monitoring tools on nodes to be added to the cluster
- Add nodes to the GPFS cluster
- Configure licenses
- Create NSDs (if any new NSDs are defined in the cluster definition)
- Run post-install environment checks

If all nodes in the cluster definition are in a cluster

- Run pre-install environment checks
- Skip all steps until NSD creation
- Create NSDs (if any new NSDs are defined in the cluster definition)
- Run post-install environment checks

Note: Although not part of GPFS, NTP configuration on every node is useful for cluster operation and future debugging. You can use the **./spectrumscale ntp** options for configuring NTP on all nodes at installation time. For more information, see **spectrumscale command** in *IBM Spectrum Scale: Administration and Programming Reference*.

What to do next

Upon completion of the installation, you will have an active GPFS cluster. Within the cluster, NSDs may have been created, performance monitoring will have been configured, and all product licenses will have been accepted. File systems will be fully created in the next step: deployment.

Install can be re-run in the future to:

- add NSD server nodes
- add GPFS client nodes
- add GUI nodes
- add NSDs
- define new file systems for use with deploy

Deploying protocols

Deployment of protocol services is performed on a subset of the cluster nodes that have been designated as protocol nodes.

Protocol nodes will have an additional set of packages installed that allow them to run the NFS, SMB and Object protocol services.

Data is served via these protocols from a pool of addresses designated as Export IP addresses or CES "public" IP addresses. The allocation of addresses in this pool will be managed by the cluster, and IP addresses will be automatically migrated to other available protocol nodes in the event of a node failure.

Before deploying protocols, there must be a GPFS cluster that has GPFS started and at least one file system for the CES shared file system.

Notes:

1. Only nodes running RHEL 7.0 and 7.1 on x86_64 and ppc64 architectures can be designated as protocol nodes.
2. The packages for all protocols will be installed on every node designated as a protocol node; this is done even if a service is not enabled in your configuration.
3. Services are enabled and disabled cluster wide; this means that every protocol node will serve all enabled protocols.
4. If SMB is enabled, the number of protocol nodes is limited to 16 nodes.
5. The **spectrumscale** installation toolkit no longer supports adding protocol nodes to an existing ESS cluster prior to ESS version 3.5.

Defining a shared file system

To use protocol services, a shared file system must be defined. If the install toolkit is used to install GPFS, NSDs can be created at this time and, if associated with a file system, the file system will then be created during deployment. If GPFS has already been configured, the shared file system can be specified manually or by re-running the **spectrumscale install** command to assign an existing NSD to the file system. If re-running **spectrumscale install**, be sure that your NSD servers are compatible with the **spectrumscale** installation toolkit and contained within the `clusterdefinition.txt` file.

Note: Be aware that if you do not configure a protocol's shared file system, the install toolkit will automatically create one for you named `ces_shared` mounted at `/ibm/ces_shared`. This will work only if you have created at least one NSD and that NSD is not already assigned to a file system.

The **spectrumscale config protocols** command can be used to define the shared file system (**-f**) and mount point (**-m**):

```
usage: spectrumscale config protocols [-h] [-l] [-f FILESYSTEM]
                                      [-m MOUNTPOINT]
```

For example: **$ ./spectrumscale config protocols -f cesshared -m /gpfs/cesshared**.

To show the current settings, issue this command:

```
$ ./spectrumscale config protocols --list
[ INFO  ] No changes made. Current settings are as follows:
[ INFO  ] Shared File System Name is cesshared
[ INFO  ] Shared File System Mountpoint is /gpfs/cesshared
```

Adding nodes to the cluster definition file

To deploy protocols on nodes in your cluster, they must be added to the cluster definition file as protocol nodes.

Run the following command to designate a node as a protocol node:

```
./spectrumscale node add NODE_IP -p
```

Enabling protocols

In order to enable or disable a set of protocols, the **spectrumscale enable** and **spectrumscale disable** commands should be used. For example:

```
$ ./spectrumscale enable smb nfs
[ INFO  ] Enabling SMB on all protocol nodes.
[ INFO  ] Enabling NFS on all protocol nodes.
```

The current list of enabled protocols is shown as part of the **spectrumscale node list** command output; for example:

```
$ ./spectrumscale node list
[ INFO  ] List of nodes in current configuration:
[ INFO  ] [Installer Node]
[ INFO  ] 9.71.18.169
[ INFO  ]
[ INFO  ] [Cluster Name]
[ INFO  ] ESDev1
[ INFO  ]
[ INFO  ] [Protocols]
[ INFO  ] Object : Disabled
[ INFO  ] SMB : Enabled
[ INFO  ] NFS : Enabled
[ INFO  ]
[ INFO  ] GPFS Node                    Admin  Quorum  Manager  NSD Server  Protocol
[ INFO  ] ESDev1-GPFS1                   X       X       X                    X
[ INFO  ] ESDev1-GPFS2                                   X                    X
[ INFO  ] ESDev1-GPFS3                                   X                    X
[ INFO  ] ESDev1-GPFS4                   X       X       X          X
[ INFO  ] ESDev1-GPFS5                   X       X       X          X
```

Configuring Object

If the object protocol is enabled, further protocol-specific configuration is required; these options are configured using the **spectrumscale config object** command, which has the following parameters:

```
usage: spectrumscale config object [-h] [-l] [-f FILESYSTEM] [-m MOUNTPOINT]
                                   [-e ENDPOINT] [-o OBJECTBASE]
                                   [-i INODEALLOCATION] [-t ADMINTOKEN]
                                   [-au ADMINUSER] [-ap ADMINPASSWORD]
                                   [-SU SWIFTUSER] [-sp SWIFTPASSWORD]
                                   [-dp DATABASEPASSWORD]
                                   [-mr MULTIREGION] [-rn REGIONNUMBER]
                                   [-s3 {on,off}]
```

The object protocol requires a dedicated fileset as its back-end storage; this fileset is defined using the **--filesystem** (**-f**), **--mountpoint**(**-m**) and **--objectbase** (**-o**) flags to define the file system, mount point, and fileset respectively.

The **--endpoint**(**-e**) option specifies the host name that will be used for access to the file store. This should be a round-robin DNS entry that maps to all CES IP addresses; this will distribute the load of all keystone and object traffic that is routed to this host name. Therefore, the endpoint is an IP address in a DNS or in a load balancer that maps to a group of export IPs (that is, CES IPs that were assigned on the protocol nodes) .

The following user name and password options specify the credentials used for the creation of an admin user within Keystone for object and container access. The system will prompt for these during **spectrumscale deploy** pre-check and **spectrumscale deploy** if they have not already been configured by

spectrumscale. The following example shows how to configure these options to associate user names and passwords: **./spectrumscale config object -au -admin -ap -dp**

The **-ADMINUSER**(**-au**) option specifies the admin user name.

The **-ADMINPASSWORD**(**-ap**) option specifies the password for the admin user.

Note: You will be prompted to enter a Secret Encryption Key which will be used to securely store the password. Choose a memorable pass phrase which you will be prompted for each time you enter the password.

The **-SWIFTUSER**(**-su**) option specifies the Swift user name.

The **-SWIFTPASSWORD**(**-sp**) option specifies the password for the Swift user.

Note: You will be prompted to enter a Secret Encryption Key which will be used to securely store the password. Choose a memorable pass phrase which you will be prompted for each time you enter the password

The **-DATABASEPASSWORD**(**-dp**) option specifies the password for the object database.

Note: You will be prompted to enter a Secret Encryption Key which will be used to securely store the password. Choose a memorable pass phrase which you will be prompted for each time you enter the password

The **-MULTIREGION**(**-mr**) option enables the multi-region object deployment feature. The **-REGIONNUMBER**(**-rn**) option specifies the region number.

The **-s3** option specifies whether the S3 (Amazon Simple Storage Service) API should be enabled.

-t ADMINTOKEN sets the `admin_token` property in the `keystone.conf` file which allows access to Keystone by token value rather than user/password. When installing with a local Keystone, by default the installer dynamically creates the `admin_token` used during initial configuration and deletes it when done. If set explicitly with `-t`, `admin_token` is not deleted from `keystone.conf` when done. The admin token can also be used when setting up a remote Keystone server if that server has `admin_token` defined.

Attention: If you want to use SELinux in the Enforcing mode, you must take that decision before proceeding with the deployment. Changing the SELinux mode after the deployment is not supported.

Adding export IPs

Note: This is mandatory for protocol deployment.

Export IPs or CES "public" IPs are used to export data via the protocols (NFS, SMB, Object). File and Object clients use these public IPs to access data on GPFS file systems. Export IPs are shared between all protocols and are organized in a public IP pool (there can be fewer public IPs than protocol nodes).

Note: Export IPs must have an associated hostname and reverse DNS lookup must be configured for each.

1. To add Export IPs to your cluster, run either this command:

   ```
   $ ./spectrumscale config protocols --export-ip-pool EXPORT_IP_POOL
   ```

   Or this command:

   ```
   $ ./spectrumscale config protocols -e EXPORT_IP_POOL
   ```

   Within these commands, `EXPORT_IP_POOL` is a comma-separated list of IP addresses.
2. To view the current configuration, run the following command:

```
$ ./spectrumscale node list
```

To view the CES shared root and the IP pool, run the following command:

```
$ ./spectrumscale config protocols -l
```

To view the Object configuration, run the following command:

```
$ ./spectrumscale config object -l
```

Running the spectrumscale deploy command

After adding the previously-described protocol-related definition and configuration information to the cluster definition file you can deploy the protocols specified in that file.

To perform deploy checks prior to deploying, use the **spectrumscale deploy** command with the **--pr** argument:

```
./spectrumscale deploy --pr
```

This is not required, however, because **spectrumscale deploy** with no argument will also run this.

Use the following command to deploy protocols:

```
spectrumscale deploy
```

Note: You will be prompted for the Secret Encryption Key that you provided while configuring object and/or authentication unless you disabled prompting.

This does the following:

- Performs pre-deploy checks.
- Creates file systems and deploys protocols as specified in the cluster definition file.
- Performs post-deploy checks.

You can explicitly specify the **--precheck** (**-pr**) option to perform a dry run of pre-deploy checks without starting the deployment. Alternatively, you can specify the **--postcheck** (**-po**) option to perform a dry run of post-deploy checks without starting the deployment. These options are mutually exclusive.

After a successful deployment, you can verify the cluster and CES configuration by running this command:

```
$ /usr/lpp/mmfs/bin/mmlscluster --ces
```

What to do next

Upon completion of the tasks described in this topic, you will have deployed additional functionality to an active GPFS cluster. This additional functionality may consist of file systems, protocol nodes, specific protocols, and authentication. Although authentication can be deployed at the same time as protocols, we have separated the instructions for conceptual purposes. Continue with the setting up authentication step if you have not yet set up authentication. See the topic “Setting up authentication” for instructions on how to set up authentication.

You can rerun the **spectrumscale deploy** command in the future to do the following:

- add file systems
- add protocol nodes
- enable additional protocols
- configure and enable authentication

Setting up authentication

Use these instructions to set up authentication for protocols.

Setting authentication configuration settings in the cluster definition file

You need to set up authentication methods for both the file and object users. If the object protocol is enabled on the protocol nodes, the installer automatically sets up the default local object authentication for object access. If you plan to configure object authentication with an external Keystone server and you are using the installation toolkit, do not configure external Keystone with the installation toolkit. For more information, see *Configuring object authentication with external Keystone server* in *IBM Spectrum Scale: Administration and Programming Reference*. To make any more changes for authentication, issue the **spectrumscale auth** command as shown in the following example:

```
$ ./spectrumscale auth -h
usage: spectrumscale auth [-h] {commitsettings,file,object} ...
```

This is a command to be run separately for Object and File in the case that your cluster has two separate servers controlling different node groups. Run the **auth** command with the data access method you wish to use and the backend server you will configure.

Authentication prerequisites:

There are a few additional prerequisites needed if you wish to configure authentication.

The following packages must be installed on all protocol nodes before running **./spectrumscale deploy**

If Object authentication is required:

- `openldap-clients`

If File authentication is required:

- `sssd`
- `ypbind`
- `openldap-clients`

To set up object authentication by using the installation toolkit:

Note: If you plan to configure object authentication with an external Keystone server and you are using the installation toolkit, do not configure external Keystone with the installation toolkit. For more information, see *Configuring object authentication with external Keystone server* in *IBM Spectrum Scale: Administration and Programming Reference*.

Object **auth** has two extra options to enable https or to enable pki. If you wish to set up either of these, you can include them in the command and you will be prompted in the next step to give the paths to the certificates required.

1. To set up object authentication, run this command:

   ```
   $ ./spectrumscale auth object [-h] [--https] [--pki] {local,external,ldap,ad}
   ```

   This will automatically open a template file for you to fill with the required **auth** settings. For more information about these settings, see the **mmuserauth** command.
2. Save the file and close it, and the settings will automatically be loaded for the installer to set up object authentication after protocols have been enabled.

   If this **auth** command has been run, authentication will automatically be enabled by the installer.

   Note: Using unusual characters or white space in settings will require you to enter the setting in single quotes (' '). For example:

   ```
   unixmap_domains = 'testdomain(1000-3000)'
   bind_username = 'My User'
   ```
3. If required, configure file authentication by following the steps provided in the next section.
4. Issue the **./spectrumscale deploy -pr** command to initiate a pre-check to make sure that the cluster is in a good state for deployment.

5. After the successful pre-check, issue the **./spectrumscale deploy** command to deploy the new authentication configuration.

To set up file authentication by using the installation toolkit:

1. To set up file authentication, run this command:

   ```
   $ ./spectrumscale auth file [-h] {ldap,ad,nis,none}
   ```

   This will automatically open a template file for you to fill with the required **auth** settings. For more detail on these settings refer to the **mmuserauth** command.
2. Save the file and close it; the settings will automatically be loaded for the installer to set up file authentication after protocols have been enabled.

   Note: Using unusual characters or white space in settings will require you to enter the setting in single quotes (' '). For example:

   ```
   unixmap_domains = 'testdomain(1000-3000)'
   bind_username = 'My User'
   ```
3. If required, configure object authentication by following the steps that are explained in the previous section.
4. Issue the **./spectrumscale deploy -pr** command to initiate a pre-check to make sure that the cluster is in a good state for deployment.
5. After the successful pre-check, issue the **./spectrumscale deploy** command to deploy the new authentication configuration.

To clear authentication settings listed in the install toolkit:

To clear authentication settings in the install toolkit, run this command:

```
$ ./spectrumscale auth clear
```

This does not clear or change a live and running authentication configuration. The **./spectrumscale auth clear** command just clears the authentication settings from the `clusterdefinition.txt` file that is used by the installation toolkit during the deployment.

Note: If the **spectrumscale** installer is used to set up object support and file support (especially SMB) with AD or LDAP Authentication, the authentication setup might cause a temporary monitoring failure and trigger an IP failover. This might lead to an error message similar to the following when configuring object : `"mmcesobjcrbase: No CES IP addresses are assigned to this node."`

If the **spectrumscale** installer failed because of this problem, do the following:

1. Check the cluster state by running the **mmlscluster --ces** command, and wait till the failed state of all nodes is cleared (`flags=none`).
2. Rebalance the IP addresses by running this command: **mmces address move --rebalance**
3. Rerun the **spectrumscale** installer to complete the object setup.

Installation of Performance Monitoring tool using the installation toolkit

By default, the Performance Monitoring tool gets installed when IBM Spectrum Scale is installed on the system using the **./spectrumscale install** command.

For more information on the Performance Monitoring tool, see *Performance Monitoring tool overview* in *IBM Spectrum Scale: Advanced Administration Guide*.

The following configuration occurs on the system after the installation:

- Sensors are installed and started on every node.
- The collector is installed and configured on two nodes in all cases apart from the following:
  - One node if there is a one-node cluster.
  - Three nodes when there are three GUI servers specified in the cluster definition.

- The proxies for the installed protocols are started. In the case of NFS and SMB, the proxy is installed with the code. In the case of object it is installed as a separate step as the final part of the installation.

During the install toolkit prechecks if is it found that the collector nodes require reconfiguring then the **./spectrumscale config perfmon -r on** CLI will need to be run to enable the install toolkit to perform this reconfiguration.

Note: A reconfiguration will result in the removal of an existing collector. To disable reconfigure run the **./spectrumscale config perfmon -r off** command:

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

Optional arguments:

-h --help show this help message and exit

-l --list List the current settings in the configuration

-r {*on*,*off*}
--enable-reconfig {on,off}

Specify if the install toolkit can reconfigure Performance Monitoring.

If reconfigure is disabled then the install toolkit will perform all installation tasks minus Performance Monitoring and the Graphical User Interface (GUI) if GUI servers are specified.

For information on how to install the Performance Monitoring tool manually, see "Manually installing the Performance Monitoring tool" on page 139.

For information on configuring the Performance Monitoring tool, see *Configuring the Performance Monitoring tool* in *IBM Spectrum Scale: Advanced Administration Guide*.

Logging and debugging

The **spectrumscale** installation toolkit reports progress to the console, and certain options also log output to a file.

Console output

The install toolkit reports progress to the console. By default only information level messages are displayed. If more verbose output is required the **-v** flag can be passed to the **spectrumscale** command.

Note: this flag must be the first argument passed to the **spectrumscale** command; for example:

```
./spectrumscale -v install --precheck
```

Log files

In addition to console output, the **install** and **deploy** functions of the **spectrumscale** installation toolkit will log verbose output to the `logs` subdirectory of the toolkit (the full log path will be reported to the console when logging begins). The log file will always contain verbose output even if the **-v** flag is not specified on the command line.

Install and deploy checks

The **install** and **deploy** functions each have a set of checks performed before and after their main action. These checks can be triggered separately by using either the **--precheck** or **--postcheck** flag.

It is safe to run the pre- and post-checks multiple times, as they will interrogate the cluster but will not make any changes.

Note that, as with the full **install** and **deploy** functions, the pre- and post-checks will create a detailed log file that can be used to provide further detail about errors encountered.

Gathering support information

If an unresolvable error is encountered when using the toolkit, a support package can be gathered using the **installer.snap.py** script (available in the same directory as the **spectrumscale** tool).

Note: The **installer.snap.py** must be run from the toolkit directory.

The **installer.snap.py** will gather information from each of the nodes defined in the configuration and will package them so they are ready for submission to the IBM Support Center. The script gathers:

- Cluster configuration file
- Install toolkit logs
- Information from the local node of the protocol services
- A GPFS snap file from each node defined in the configuration
- Basic environment information from the local node including:
  - Current OS and kernel version
  - System message log (`dmesg`)
  - File system usage
  - Network configuration
  - Current processes

Collection of core dump data

See the topic about changing configurations to enable **spectrumscale** to collect core dump data in "Understanding the **spectrumscale** installation toolkit options" on page 112.

Upgrading GPFS components with the spectrumscale installation toolkit

You can use the **spectrumscale** installation toolkit to upgrade all components (GPFS, NFS, Object, SMB, and the Performance Monitoring tool).

Important: If you have all protocols enabled, plan on a system down time of approximately 20 minutes per protocol node.

Before using the **spectrumscale** installation toolkit to perform upgrades, do the following:

1. Go to the IBM Spectrum Scale page on Fix Central, select the new **spectrumscale** package and then click **Continue**.
2. Choose the download option **Download using Download Director** to download the new **spectrumscale** package and place it in the wanted location on the install node.

   Note: Although it is not recommended but if you must use the download option **Download using your browser (HTTPS)**, instead of clicking the down arrow to the left of the package name, you must right-click on the package name and select the **Save Link As..** option. If you click the download arrow, the browser might hang.
3. Extract the new **spectrumscale** self-extracting package by running the package name (for example, `/tmp/Spectrum_Scale_Protocols_Standard-4.2.0.0_x86_64-Linux_install`).

   This creates a new directory structure (`/usr/lpp/mmfs/4.2.0.0/`).
4. Accept the license agreement. The system will guide you through next steps including:
   - cluster installation and protocol deployment
   - upgrading an existing cluster using the install toolkit
   - adding nodes to an existing cluster using the install toolkit

- adding NSDs or file systems to an existing cluster using the install toolkit

5. If you are upgrading protocol nodes you need to stop the Performance Monitoring tool Object proxy service. Depending on the version of the Performance Monitoring tool you have installed, it will either be `pmprovider.service` or `pmswiftd.service`.

   They can be stopped by issuing the following commands:

   - **`systemctl stop pmswiftd.service`**
   - **`systemctl stop pmprovider.service`**

6. Determine which of the following situations applies to you:
   - If you already have GPFS installed and did not use the toolkit to do so, but you would like to upgrade using the toolkit, follow these steps:
     a. Configure your cluster definition file as explained in the following topics:
        – "Defining configuration options for the spectrumscale installation toolkit" on page 115
        – "Setting up the install node" on page 115
     b. Proceed to the "spectrumscale upgrade command" section that follows.
   - If you have GPFS installed and have already used the install toolkit, follow these steps:
     a. Copy the cluster definition file from your `/usr/lpp/mmfs/4.1.1.x/installer/configuration/clusterdefinition.txt` file to the newly extracted `/usr/lpp/mmfs/4.2.0.0/installer/configuration/clusterdefinition.txt` directory.
     b. Ensure that the cluster definition file is accurate and reflects the current state of your Spectrum Scale cluster.
     c. Proceed to the "spectrumscale upgrade command" section that follows.
   - If you have no existing GPFS cluster and would like to use the install toolkit to install both GPFS and an update, follow these steps:
     a. Copy the `/usr/lpp/mmfs/4.2.0.0/installer` directory (for example, `/usr/lpp/mmfs/4.2.0.0/installer` over the top of the `/usr/lpp/mmfs/4.2.0.0/installer` directory.
     b. Complete defining your cluster definition file as explained in the following topics:
        – "Defining configuration options for the spectrumscale installation toolkit" on page 115
        – "Setting up the install node" on page 115
     c. Run the **spectrumscale install** command to install GPFS. (See "Installing GPFS" on page 118.)
     d. Copy the `configuration/clusterdefinition.txt` file from the location where you just ran the **spectrumscale install** command to `4.2.0.0/installer/configuration` (for example, `4.2.0.0/installer/configuration`).
     e. Add nodes to ensure that the `gpfs.base` rpm is installed.
     f. Proceed to upgrade GPFS (for example, to 4.x.x.x); see the "The spectrumscale upgrade command" section that follows.
     g. Proceed to deploy protocols (if required) as explained in "Deploying protocols" on page 121.

The spectrumscale upgrade command

The **./spectrumscale upgrade** command is used to perform upgrades and has these options:

`-ve`
Shows the current version of install packages and the available version to upgrade to.

`-pr`
Performs health checks and ensures all prerequisites are met on the cluster prior to upgrade.

`-po`
Performs health checks on the cluster after the upgrade has been completed.

`-f` Bypasses the message that will appear when upgrading components, that warns that there may be an outage when upgrading SMB and the performance monitoring tool.

As with all **spectrumscale** commands, use the **-v** flag for verbose output. The **-v** flag must be placed immediately after **./spectrumscale**.

For example: **./spectrumscale -v upgrade**

Note: The **upgrade** command can be used to upgrade GPFS from version 4.1.0 and later.

Running upgrade prechecks

For any cluster with existing Performance Monitoring collectors, the upgrade prechecks might find that a reconfiguration is required. To enable the install toolkit to perform the reconfiguration, issue the following command:

```
./spectrumscale config perfmon -r on
```

To keep your existing algorithms, issue the following command:

```
./spectrumscale config perfmon -r off
```

Before upgrading, run upgrade prechecks by issuing the following command:

```
/usr/lpp/mmfs/4.2.0.x/installer/spectrumscale upgrade -pr
```

Upgrading all components

Note: Before upgrading all components, run the **upgrade precheck** command.

All components can be upgraded at once using the following command:

```
./spectrumscale upgrade
```

This command can still be used even if all protocols are not enabled; however, the packages for other protocols will be upgraded also (but not started).

This will cause a brief outage of the performance monitoring tool and SMB (if enabled). Read the following sections for more information about the impact of upgrade on each component.

Once upgrade is complete, run the **spectrumscale upgrade -ve** command to show the versions of the packages installed to ensure that they are all at the desired level.

Proceed to the next step, migration to a new level of GPFS. See this topic for instructions "Completing the migration to a new level of IBM Spectrum Scale" on page 178.

Failure recovery

Should a failure occur during an upgrade, examine the log at the location provided in the output from the install toolkit to determine the cause. More information on each **spectrumscale** error will be provided in the output from the install toolkit.

Certain failures will cause the upgrade process to stop. In this case, the cause of the failure should be addressed on the node on which it occurred. Once the problem has been addressed, the upgrade command can be run again. This will not affect any nodes that were upgraded successfully and will continue once the point of failure has been reached.

An example of a failure during upgrade is:

A protocol is enabled in the configuration file, but is not running on a node.
Using the **mmces service list** command on the node that failed will highlight which process is not running (the output from the install toolkit will also report which component failed). Make sure the component is started on all nodes with **mmces service start nfs | smb | obj**, or

alternatively disable the protocol in the configuration using **spectrumscale disable nfs|smb|object** if the component has been intentionally stopped.

For information on troubleshooting installation. deployment, and upgrade related issues, see *Resolving most frequent problems related to installation, deployment, and upgrade* in *IBM Spectrum Scale: Problem Determination Guide*.

Upgrade considerations for IBM Spectrum Scale for object storage

- If object authentication is configured with AD or LDAP and anonymous bind is not used during object authentication configuration, use the following command after upgrading IBM Spectrum Scale for object storage from 4.1.x to 4.2.x:

  ```
  mmobj config change --ccrfile keystone.conf --section ldap --property password --value ldapbindpassword
  ```

- After upgrading IBM Spectrum Scale for object storage, some migration steps need to be performed to enable features such as compression for existing file systems. For information about these migration steps, see "Completing the migration to a new level of IBM Spectrum Scale" on page 178.
- If the **MAX_FILE_SIZE** parameter is not changed in your setup, verify that its value is correct (5 TiB) and change it if required.
  1. Verify if the value of the **MAX_FILE_SIZE** parameter is 5497558138882 using the following command:

     ```
     mmobj config list --ccrfile swift.conf --section 'swift-constraints' \
     --property max_file_size
     ```

     The system displays the following output:

     ```
     max_file_size = 5497558138882
     ```

  2. If **MAX_FILE_SIZE** is not correct (5 TiB), set **MAX_FILE_SIZE** using the following command:

     ```
     mmobj config change --ccrfile swift.conf --section 'swift-constraints' \
     --property max_file_size --value 5497558138882
     ```

Installing IBM Spectrum Scale management GUI by using `spectrumscale` installation toolkit

Use the following information for installing the IBM Spectrum Scale management GUI using the **spectrumscale** installation toolkit.

The IBM Spectrum Scale is only installed when nodes have been specified as GUI servers in the cluster definition with the -g option:

```
./spectrumscale node add gpfsnode3 -g
```

The IBM Spectrum Scale GUI requires passwordless ssh to all other nodes in the cluster. Therefore, the GUI server node must be added as an admin node using the -a flag:

```
./spectrumscale node add gpfsnode3 -a
```

Note: It is recommended not to configure a node as both GUI management node and protocol node.

If no nodes have been specified as GUI servers, then the GUI will not be installed. It is recommended to have at least 2 GUI interface servers and a maximum of 3 for redundancy.

The GUI will be installed on specified GUI servers when you run the **./spectrumscale install** command and on protocol nodes also specified as GUI servers during the **./spectrumscale deploy** command.

At the end of a successful GPFS installation or protocol deployment, you can access the GUI through a web browser with the following node address: `https://<GUI server IP or hostname>`.

Note: The default user name and password to access the IBM Spectrum Scale management GUI is `admin` and `admin001` respectively.

Protocol node IP further configuration

The commands for adding or moving IPs are as follows.

Checking that protocol IPs are currently configured

The **mmces address list** command shows all the currently configured protocol IPs.

Note: The term CES-IP refers to the cluster export service IP, or a protocol IP used specifically for NFS, SMB, Object protocol access.

```
mmces address list
```

The system displays output similar to the following:

```
Node  Daemon node name  IP address     CES IP address list
-----------------------------------------------------------------------
  1   prt001st001       172.31.132.1    10.0.0.101
  4   prt002st001       172.31.132.2    10.0.0.102
  5   prt003st001       172.31.132.3    10.0.0.103
  6   prt004st001       172.31.132.4    10.0.0.104
  7   prt005st001       172.31.132.5    10.0.0.105
  8   prt006st001       172.31.132.6    10.0.0.106
```

Additional IPs can be added to a node. In this example, IPs `10.0.0.108` and `10.0.0.109` are added to protocol node 5, `prt005st001`.

```
mmces address add --node prt005st001 --ces-ip 10.0.0.108,10.0.0.109
mmces address list
```

The system displays output similar to the following:

```
Node  Daemon node name  IP address     CES IP address list
-----------------------------------------------------------------------
  1   prt001st001       172.31.132.1    10.0.0.101
  4   prt002st001       172.31.132.2    10.0.0.102
  5   prt003st001       172.31.132.3    10.0.0.103
  6   prt004st001       172.31.132.4    10.0.0.104
  7   prt005st001       172.31.132.5    10.0.0.105 10.0.0.108 10.0.0.109
  8   prt006st001       172.31.132.6    10.0.0.106
```

IPs can also be moved among nodes manually. This is helpful if IP allocation becomes unbalanced among the nodes. Continuing from the prior example, `prt005st001` now has three protocol IPs whereas all other nodes have a single protocol IP. To balance this out a bit better, `10.0.0.109` will be manually moved to node `prt004st001`.

```
mmces address move –ces-ip 10.0.0.109 --node prt004st001
mmces address list
```

The system displays output similar to the following:

```
Node  Daemon node name  IP address     CES IP address list
-----------------------------------------------------------------------
  1   prt001st001       172.31.132.1    10.0.0.101
  4   prt002st001       172.31.132.2    10.0.0.102
  5   prt003st001       172.31.132.3    10.0.0.103
  6   prt004st001       172.31.132.4    10.0.0.104 10.0.0.109
  7   prt005st001       172.31.132.5    10.0.0.105 10.0.0.108
  8   prt006st001       172.31.132.6    10.0.0.106
```

List all protocol services enabled

Run the following command from any protocol node to list the running services on all protocol nodes:

```
mmces service list -a
```

The system displays output similar to the following:

```
Enabled services: SMB NFS OBJ
prt001st001: SMB is running, NFS is running, OBJ is running
prt002st001: SMB is running, NFS is running, OBJ is running
prt003st001: SMB is running, NFS is running, OBJ is running
```

Object protocol further configuration

If the Object protocol was enabled during installation, use the following information to verify the Object protocol configuration and to enable features such as unified file and object access and multi-region object deployment.

Object is currently set up, enabled, and configured by the **spectrumscale** installation toolkit, but a few additional steps are necessary to ready the Object protocol for use. Verify the Object protocol by running a few tests.

Verifying the Object protocol portion of the installation

Perform the following simple example Openstack client and Swift commands to verify your installation. You should be able to list all users and projects (called accounts in Swift), list the Swift containers, upload a sample object to a container, and list that container and see the object. These commands should complete with no errors.

```
# source $HOME/openrc
# openstack user list
# openstack project list
# swift stat
# date > object1.txt
# swift upload test_container object1.txt
object1.txt
# swift list test_container
object1.txt
```

Enabling multi-region object deployment initially

For multi-region object deployment, each region is a separate cluster and on each cluster IBM Spectrum Scale for object storage is installed independently using the installer.

In a single cluster, the installer completely installs the object protocol on all the nodes within that cluster. However, in a multi-region object deployment environment, each cluster is installed independently and the object protocol on the independent clusters is linked together.

The object portion of the IBM Spectrum Scale installer contains an option to indicate if multi-region support is to be enabled. If it needs to be enabled, installer determines if the cluster being installed is the first cluster or if the cluster is a subsequent one and it will be added to an existing environment. This information is passed to the **mmobj swift base** command that the installer runs to install the object protocol.

To set up an initial multi-region environment, issue the following command on the 1st cluster after it has been installed:

```
mmobj multiregion enable
```

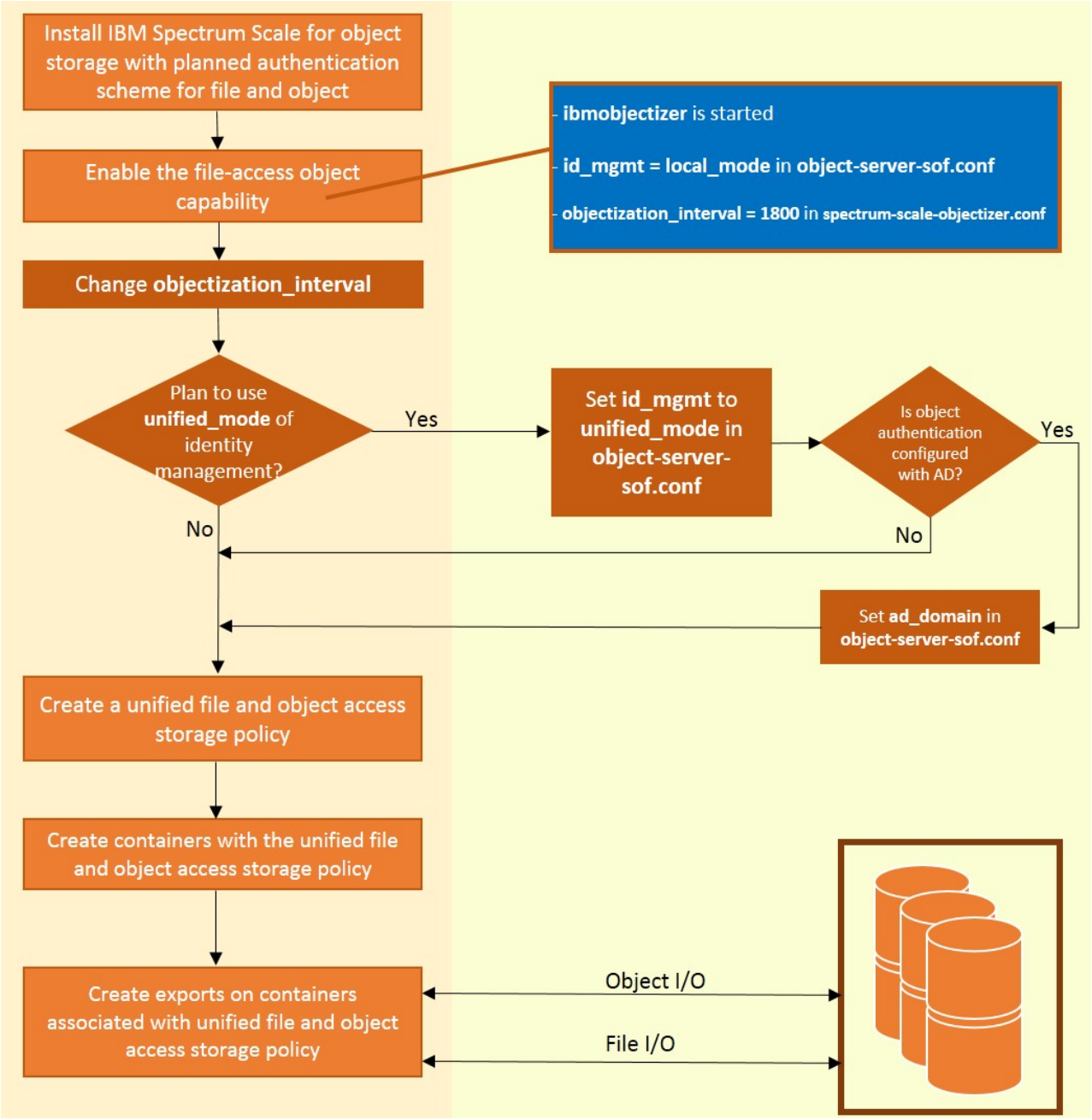
For information on adding a new region to a multi-region object deployment environment, see *Adding a region in a multi-region object deployment* in *IBM Spectrum Scale: Administration and Programming Reference*.

Installing and using unified file and object access

Unified file and object access gets installed when you install IBM Spectrum Scale for object storage in an IBM Spectrum Scale cluster. No additional steps are required for installing unified file and object access.

For information about managing and administering unified file and object access, see *Unified file and object access in IBM Spectrum Scale* in *IBM Spectrum Scale: Administration and Programming Reference*.

The high-level flow of administering unified file and object access is shown in the following diagram. The flow of changing the identity management mode for unified file and object access with the **id_mgmt** parameter is also shown.



For detailed steps, see *Administering unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

Enabling unified file and object access after migrating from IBM Spectrum Scale 4.1.1 to 4.2

Use these steps to enable unified file and object access after migrating from IBM Spectrum Scale Version 4.1.1 to 4.2.

1. Enable the file-access object capability.

   For detailed steps, see *Enabling the file-access object capability* in *IBM Spectrum Scale: Administration and Programming Reference*.
2. By default, the ibmobjectizer service interval is set to 30 minutes. Set the ibmobjectizer service interval to another value according to your requirement.

   For detailed steps, see *Setting up the objectizer service interval* in *IBM Spectrum Scale: Administration and Programming Reference*.
3. By default, the identify management mode for unified file and object access is set to `local_mode`. Change the identity management mode according to your requirement.

   For detailed steps, see *Configuring authentication and setting identity management modes for unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

For the other unified file and object access tasks that you can perform, see *Administering unified file and object access* in *IBM Spectrum Scale: Administration and Programming Reference*.

Manually installing the GPFS software packages on Linux nodes

Use these instructions to manually install the GPFS software on Linux nodes.

The GPFS software packages are installed using the **rpm** command (for SLES and Red Hat Enterprise Linux) or the **dpkg** command (for Debian Linux).

Required packages

The following packages are required for SLES and Red Hat Enterprise Linux:

- `gpfs.base-4.2.*.rpm`
- `gpfs.gpl-4.2.*.noarch.rpm`
- `gpfs.msg.en_US-4.2.*.noarch.rpm`
- `gpfs.gskit-8.0.50.*.rpm`
- `gpfs.ext-4.2.*.rpm` (IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition only)
- `gpfs.crypto-4.2.*.rpm` (IBM Spectrum Scale Advanced Edition only)
- `gpfs.adv-4.2.*.rpm` (IBM Spectrum Scale Advanced Edition only)

The following packages are required for Debian Linux:

- `gpfs.base-4.2.*.deb`
- `gpfs.gpl-4.2.*all.deb`
- `gpfs.msg.en_US-4.2.*all.deb`
- `gpfs.gskit-8.0.50.*.deb`
- `gpfs.ext-4.2.*.deb` (IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition only)
- `gpfs.crypto-4.2.*.deb` (IBM Spectrum Scale Advanced Edition only)
- `gpfs.adv-4.2.*.deb` (IBM Spectrum Scale Advanced Edition only)

Note: For this release, the IBM Global Security Kit (GSKit) version for RPMs and Debian Linux packages must be at least `8.0.50.47` or later.

Optional packages

The `gpfs.docs-4.2.*noarch.rpm` package is optional for SLES and Red Hat Enterprise Linux.

The `gpfs.docs-4.2.*all.deb` package is optional for Debian Linux.

In addition to the above listed packages, the following optional packages are also available for IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition

- `gpfs.callhome-jre-8.0-1.*.rpm`

- gpfs.callhome-4.2-1.*.rpm
- gpfs.callhome-ecc-client-4.2-1.*.rpm
- gpfs.gui-4.2.0-0*.rpm
- gpfs.gss.pmcollector-4.2.*.rpm
- gpfs.gss.pmsensors-4.2.*.rpm

The following packages are also optional for Debian Linux and Ubuntu 14.04 (x86_64-linux only).

- gpfs.gss.pmcollector-4.2.*.deb
- gpfs.gss.pmsensors-4.2.*.deb

The following packages are optional for SLES and Red Hat Enterprise Linux (x86_64-linux and ppc64-linux platforms):

- gpfs.callhome-jre-8.0-1.*.rpm
- gpfs.callhome-4.2-1.*.rpm
- gpfs.callhome-ecc-client-4.2-1.*.rpm

The following packages are required (and provided) only on the Elastic Storage Server (ESS):

- gpfs.gnr.base-1.0.0-0.ppc64.rpm
- gpfs.gnr-4.2.*.ppc64.rpm
- gpfs.gss.firmware-4.2.*.ppc64.rpm

For more information about Elastic Storage Server (ESS), see *Deploying the Elastic Storage Server*.

The gpfs.gnr-3.5.*.ppc64.rpm package is required only if GPFS Native RAID on Power 775 is used. For more information about GPFS Native RAID, see *IBM Spectrum Scale RAID: Administration*.

To install all of the GPFS SLES or Red Hat Enterprise Linux RPMs for the Spectrum Scale Express edition, issue the following command:

```
cd /usr/lpp/mmfs/4.2.0.0
```

then, issue this command:

```
rpm -ivh gpfs.base*.rpm gpfs.gpl*rpm gpfs.gskit*rpm gpfs.msg*rpm
```

To install all of the required GPFS SLES or Red Hat Enterprise Linux RPMs for the Spectrum Scale Standard edition, issue this command:

```
cd /usr/lpp/mmfs/4.2.0.0
```

then, issue this command:

```
rpm -ivh gpfs.base*.rpm gpfs.gpl*rpm gpfs.gskit*rpm gpfs.msg*rpm gpfs.ext*rpm
```

To install all of the required GPFS SLES or Red Hat Enterprise Linux RPMs for the Spectrum Scale Advanced edition, issue the following command:

```
cd /usr/lpp/mmfs/4.2.0.0
```

then, issue this command:

```
rpm -ivh gpfs.base*.rpm gpfs.gpl*rpm gpfs.gskit*rpm gpfs.msg*rpm gpfs.ext*rpm gpfs.adv*rpm gpfs.crypto*rpm
```

To install all of the required GPFS Linux on System z® packages for IBM Spectrum Scale, issue the following command:

```
cd /usr/lpp/mmfs/4.2.0.0
```

then, issue this command:

```
rpm -ivh gpfs.base*.rpm gpfs.gpl*rpm gpfs.gskit*rpm gpfs.msg*rpm gpfs.ext*rpm gpfs.adv*rpm
```

To install all of the GPFS Debian Linux packages for the IBM Spectrum Scale Express edition, issue the following command:

```
cd /usr/lpp/mmfs/4.2.0.0/gpfs*.deb
```

then, issue this command:

```
dpkg -i gpfs.base*deb gpfs.gpl*deb gpfs.gskit*deb gpfs.msg*deb
```

To install all of the GPFS Debian Linux packages for the IBM Spectrum Scale Standard edition, issue the following command:

```
cd /usr/lpp/mmfs/4.2.0.0/gpfs*.deb
```

then, issue this command:

```
dpkg -i gpfs.base*deb gpfs.gpl*deb gpfs.gskit*deb gpfs.msg*deb gpfs.ext*deb
```

To install all of the GPFS Debian Linux packages for the IBM Spectrum Scale Advanced edition, issue the following command:

```
cd /usr/lpp/mmfs/4.2.0.0/gpfs*.deb
```

then, issue this command:

```
dpkg -i gpfs.base*deb gpfs.gpl*deb gpfs.gskit*deb gpfs.msg*deb gpfs.ext*deb gpfs.adv*deb gpfs.crypto*deb
```

To install optional gpfs.doc package for GPFS SLES or Red Hat Enterprise Linux issue the following command:

```
rpm -ivh /usr/lpp/mmfs/4.2.0.0/gpfs.doc*rpm
```

To install optional gpfs.doc package for GPFS Debain Linux package, issue the following command:

```
dpkg -i /usr/lpp/mmfs/4.2.00/gpfs.doc*deb
```

To install the GPFS GUI, see “Manually installing IBM Spectrum Scale management GUI” on page 144.

For information on installing the call home feature, see *IBM Spectrum Scale: Advanced Administration Guide*the *Monitoring* section.

See also the following topics:

“Manually installing the Performance Monitoring tool” on page 139

“Object protocol further configuration” on page 133

Verifying the GPFS installation on Debian Linux nodes

You can verify the installation of the GPFS Debian Linux packages on each node.

To check that the software has been successfully installed, use the **dpkg** command:

```
dpkg -l | grep gpfs
```

The system returns output similar to the following:

```
ii gpfs.base 4.2.0-0      GPFS File Manager
ii gpfs.docs 4.2.0-0      GPFS Server Man Pages and Documentation
ii gpfs.gpl 4.2.0-0       GPFS Open Source Modules
ii gpfs.gskit 8.0.50.47   GPFS GSKit Cryptography Runtime
ii gpfs.msg.en_US 4.2.0-0 GPFS Server Messages - U.S. English
```

If you have the IBM Spectrum Scale Standard Edition or the IBM Spectrum Scale Advanced Edition installed, you should also see the following line in the output:

```
ii gpfs.ext 4.2.0-0       GPFS Extended Features
```

If you have the IBM Spectrum Scale Advanced Edition installed, you should also see the following line in the output:

```
ii gpfs.crypto 4.2.0-0    GPFS Cryptographic Subsystem
ii gpfs.adv 4.2.0-0    GPFS Advanced Features
```

Verifying the GPFS installation on SLES and Red Hat Enterprise Linux nodes

You can verify the installation of the GPFS SLES or Red Hat Enterprise Linux RPMs on each node.

To check that the software has been successfully installed, use the **rpm** command:

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

The system returns output similar to the following:

```
gpfs.docs-4.2.0-0
gpfs.base-4.2.0-0
gpfs.msg.en_US-4.2.0-0
gpfs.gpl-4.2.0-0
gpfs.gskit-8.0.50.47
```

If you have the IBM Spectrum Scale Standard Edition or the IBM Spectrum Scale Advanced Edition installed, you should also see the following line in the output:

```
gpfs.ext-4.2.0-0
```

If you have the IBM Spectrum Scale Advanced Edition installed, you should also see the following in the output:

```
gpfs.crypto-4.2.0-0
gpfs.adv-4.2.0-0
```

For installations that include GPFS Native RAID, you should also see the following line in the output:

```
gpfs.gnr-4.2.0-0
```

For more information about GPFS Native RAID, see *IBM Spectrum Scale RAID: Administration*.

Manually installing IBM Spectrum Scale for object storage

IBM Spectrum Scale for object storage is typically installed using the spectrumscale installation toolkit. If you do not want to use the spectrumscale installation toolkit, use the following steps to manually install IBM Spectrum Scale for object storage.

Before you begin manually installing IBM Spectrum Scale for object storage, complete the following prerequisite tasks.

1. Create protocol nodes for object service. For more information, see *CES and protocol configuration* in *IBM Spectrum Scale: Administration and Programming Reference*.
2. Add at least one CES IP address.
3. On all protocol nodes, install the `spectrum-scale-object` package and its associated dependencies. The package is created in the `/usr/lpp/mmfs/4.2.0.0/object_rpms` directory by expanding the `Spectrum_Scale_Protocols` installation image. For more information about extracting an installation image, see "Extracting the GPFS software on Linux nodes" on page 103.

   You can also install the package using the following command:

   ```
   yum -y install spectrum-scale-object
   ```

   You might need to create the **yum** repository before using this command. To create the **yum** repository, create an entry similar to the following in the `/etc/yum.repos.d` directory:

```
[spectrum_scale]
                    name=spectrum_scale
                    baseurl=file:///usr/lpp/mmfs/4.2.0.0/object_rpms
                    enabled=1
```

Manually install the object protocol and then enable object services as follows.

1. From one of the protocol nodes, install the object protocol by using the **mmobj swift base** command.

   For example:

   ```
   # mmobj swift base -g /gpfs/fs1 -o object_fileset  \
   --cluster-hostname protocol-cluster.example.net  --local-keystone --admin-password passw0rd
   mmobj swift base: Validating execution environment.
   mmobj swift base: Performing SELinux configuration.
   mmobj swift base: Creating fileset /dev/fs1 object_fileset.
   mmobj swift base: Configuring Keystone server in /gpfs/fs1/object/keystone.
   mmobj swift base: Creating postgres database.
   mmobj swift base: Validating Keystone environment.
   mmobj swift base: Validating Swift values in Keystone.
   mmobj swift base: Configuring Swift services.
   mmobj swift base: Uploading configuration changes to the CCR.
   mmobj swift base: Configuration complete.
   ```

After the initial install is complete, the object protocol needs to be enabled across all the protocol nodes.

2. Complete the configuration and start object services on all protocol nodes by using the **mmces service enable** command.

   ```
   # mmces service enable OBJ
   ```

Manually installing the Performance Monitoring tool

The Performance Monitoring tool is automatically installed by the **spectrumscale** installation toolkit. You can also install the Performance Monitoring tool manually for the releases that are not supported by the installation toolkit.

For more information, see "Understanding the **spectrumscale** installation toolkit options" on page 112. For more information about the Performance Monitoring tool, see *Configuring the Performance Monitoring tool* in *IBM Spectrum Scale: Advanced Administration Guide*.

During an upgrade to the tool or in case of any issues, you can manually install the tool by performing the following steps:

1. Download and install the following packages, which are available in `/usr/lpp/mmfs/4.2.0.0/zimon_rpms`.

   ```
   gpfs.gss.pmsensors_version-release.os.target_arch.file_format
   gpfs.gss.pmcollector_version-release.os.target_arch.file_format
   ```

   Then use your operating system's native package management mechanism.

   For example:

   To install 4.2.0-0 sensors on a Red Hat 7, x86_64 node use the following command:

   ```
   rpm -ivh gpfs.gss.pmsensors_4.2.0-0.el7.x86_64.rpm
   ```

   To install 4.2.0-0 sensors on a Ubuntu 14, amd64 node use the following command:

   ```
   dpkg -i gpfs.gss.pmsensors_4.2.0-0.U14.04_amd64.deb
   ```

   To install a 4.2.0-0 collector on a Red Hat 7, ppc64 little endian node use the following command:

   ```
   rpm -ivh gpfs.gss.pmcollector-4.2.0-0.el7.ppc64le.rpm
   ```

   To install a 4.2.0-0 collector on a Red Hat 6, ppc64 node use the following command:

   ```
   rpm -ivh gpfs.gss.pmcollector-4.2.0-0.el6.ppc64.rpm
   ```

2. A single collector can easily support up to 400 sensor nodes. The collector can be any node on the system. All sensors will report to this node. Select any node in the system to be the collector node and modify the /opt/IBM/zimon/ZIMonSensors.cfg file on the node as follows:
   - If you are running the NFS protocol and want NFS metrics then include the following:
     ```
     {
         # NFS Ganesha statistics
         name = "NFSIO"
         period = 1
         type = "Generic"
     },
     ```
   - If you are running the SMB protocol and want SMB metrics then include the following:
     ```
     {
             name = "SMBStats"
             period = 1
             type = "Generic"
     },
     {
             name = "SMBGlobalStats"
             period = 1
             type = "Generic"
     },
     {
             name = "CTDBStats"
             period = 1
             type = "Generic"
     },
     {
             name = "CTDBDBStats"
             period = 1
             type = "Generic"
     },
     ```
   - At the bottom of the file add the following:
     ```
     collectors =
     {
             host = "<ip of collector node>|<fully qualified domain name of collector node>"
             port = "4739"
     }
     ```
   - To enable performance monitoring for Object, install the pmswift rpm:

     `rpm -ivh pmswift-<version>-<release>.noarch.rpm`

     where <version> is equal to or greater than 4.2 and <release> is equal to or greater than 0.

     The installation of the pmswift rpm also copies SWIFT related sensors configuration files, namely, SwiftAccount.cfg, SwiftContainer.cfg, SwiftObject.cfg and SwiftProxy.cfg to the Performance Monitoring tool's installation directory, /opt/IBM/zimon/. The pmswift rpm converts the operational metrics for Object into a form that is usable by the Performance Monitoring tool.

     After installation of the pmswift rpm, the following steps must be carried out:

     a. Edit the Object configuration files for all Object servers that reside in cluster configuration repository (CCR), using the following command:

        `/usr/local/pmswift/bin/pmswift-config-swift set`

        CCR will then propagate modified configuration files to /etc/swift/ directory on all the protocol nodes within the cluster. The modified configuration files are:

        - account - *.conf
        - container - *.conf
        - object - *.conf
        - proxy - *.conf

b. Use the /usr/local/pmswift/bin/pmswift-config-zimon set command to edit the sensors configuration information stored in the CCR. This adds the SWIFT related following sensors entries:

```
{
        # SwiftAccount operational metrics
        name = "SwiftAccount"
        period = 1
        type = "generic"
},
{
        # SwiftContainer operational metrics
        name = "SwiftContainer"
        period = 1
        type = "generic"
},
{
        # SwiftObject operational metrics
        name = "SwiftObject"
        period = 1
        type = "generic"
},
{
        # SwiftProxy operational metrics
        name = "SwiftProxy"
        period = 1
        type = "generic"
},
```

These entries are then automatically propagated to the ZIMonSensors.cfg file in /opt/IBM/zimon on all the nodes in the cluster.

c. Start the pmswiftd.service using the following command:

```
systemctl start pmswiftd.service
```

d. Start/restart the pmsensors.service using the following command:

```
systemctl start|restart pmsensors.service
```

For more information on how to manually upgrade pmswift, see the "Manually upgrading pmswift" on page 142 topic.

3. If the protocol sensors are enabled on a GPFS-only node, you will see an error regarding them being unavailable, however, the other sensors will continue running.
4. Start the sensors on each node using the **systemctl start pmsensors.service** command.
5. On the collector nodes, start the collector, using the **systemctl start pmcollector.service** command.
6. To ensure that sensors and collectors are restarted after the node reboots, you can enable them using the following commands:

Sensors

To disable sensors, use the **systemctl disable pmsensors.service** command.

To enable sensors, use the **systemctl enable pmsensors.service** command.

Collector

To disable the collector, use the **systemctl disable pmcollector.service** command.

To enable the collector, use the **systemctl enable pmcollector.service** command.

The collector node will start gathering all the requested metrics.

Note: Although you can enable sensors on every node in a system, do note that with the increase in number of nodes, the metric collection work for the collector also increases. It is recommended to ensure that collection of metrics does not increase above 1000000 metrics per second.
By default, the install toolkit enables metrics on each protocol node but not on the GPFS nodes.

Metrics can be retrieved from any node in the system using the **mmperfmon query** command. For more information, see *mmperfmon command* in *IBM Spectrum Scale: Administration and Programming Reference*.

For more information about the Performance Monitoring tool, see *Configuring the Performance Monitoring tool* in *IBM Spectrum Scale: Advanced Administration Guide*.

Manually upgrading pmswift

To upgrade pmswift you can either uninstall and reinstall the pmswift rpm or use the native rpm upgrade command: `rpm -Uvh pmswift-version-release.noarch.rpm`. After upgrading, restart the service using the following command: `systemctl restart pmswiftd.service`.

Uninstall pmswift-version-release.noarch.rpm

1. Stop the pmsensors.service using the following command:
   ```
   systemctl stop pmsensors.service
   ```
2. If uninstalling pmswift-4.1.1-4 or later, stop the pmswiftd.service using the following command:
   ```
   systemctl stop  pmswiftd.service
   ```

   If uninstalling pmswift-4.1.1-3, stop the pmprovidor.service using the following command:
   ```
   systemctl stop pmprovider.service
   ```
3. Uninstall the pmswift rpm using the following command:
   ```
   rpm -evh --nodeps pmswift
   ```

   If you are uninstalling pmswift-4.1.1-3, it should edit the Object configuration files for all Object servers and remove the entries created at the time of installation. The Object configuration files in `/etc/swift/ directory` are:
   - account - *.conf
   - container - *.conf
   - object - *.conf
   - proxy - *.conf

   This should also edit the sensors configuration file, `/opt/IBM/zimon/ZIMonSensors.cfg`, to remove the Object related sensors entries created at the time of installation. If you are uninstalling pmswift-4.1.1-4 or later these files will be left alone.
4. Ensure that following directories/files are removed. If they are not removed, you can, remove them manually.
   a. `/usr/local/swiftmon` directory or `/usr/local/pmswift` directory
   b. `/var/log/swiftmon` directory or `/var/log/pmswift` directory
   c. `/var/run/swiftmon` directory or `/var/runpmswift.pid` file
   d. For pmswift-4.1.1-4 and later remove `/etc/rc.d/init.d/pmswift` file and for pmswift-4.1.1-3 remove `/etc/rc.d/init.d/pmprovider` file
   e. For pmswift-4.1.1-3 `SwiftAccount.cfg`, `SwiftContainer.cfg`, `SwiftObject.cfg` and `SwiftProxy.cfg` files from within the Performance Monitoring tool's installation directory, `/opt/IBM/zimon/`.
5. Ensure that for pmswift-4.1.1-3 the pmprovider.service and for pmswift-4.1.1-4 and later the pmswftd.service is not available anymore by running the following command:
   ```
   systemctl daemon-reload
   ```

Install pmswift-version-release.noarch.rpm

1. Install the pmswift rpm using the following command:
   ```
   rpm -ivh pmswift-version-release.noarch.rpm
   ```
2. Ensure that following directories/files have been created:
   a. `/usr/local/pmswift` directory

b. /var/log/pmswift directory
c. /etc/logrotate.d/pmswift file
d. /etc/rc.d/init.d/pmswiftd file
e. SwiftAccount.cfg, SwiftContainer.cfg, SwiftObject.cfg and SwiftProxy.cfg files in the Performance Monitoring tool's installation directory, /opt/IBM/zimon/.

3. Edit the Object configuration files for all Object servers that reside in CCR, using the /usr/local/pmswift/bin/pmswift-config-swift set command. CCR will then propagate modified configuration files to /etc/swift/ directory on all the protocol nodes within the cluster. The modified configuration files are:
   - account - *.conf
   - container - *.conf
   - object - *.conf
   - proxy - *.conf
4. Edit the sensors configuration file information stored in the CCR using the /usr/local/pmswift/bin/pmswift-config-zimon set command to add the following Object related sensors entries:

```
{
  # SwiftAccount operational metrics
  name = "SwiftAccount"
  period = 1
  type = "generic"
  restrict= "cesNodes"
},
{
  # SwiftContainer operational metrics
  name = "SwiftContainer"
  period = 1
  type = "generic"
  restrict= "cesNodes"
},
{
  # SwiftObject operational metrics
  name = "SwiftObject"
  period = 1
  type = "generic"
  restrict= "cesNodes"
},
{
  # SwiftProxy operational metrics
  name = "SwiftProxy"
  period = 1
  type = "generic"
  restrict= "cesNodes"
},
```

   These entries are then automatically propagated to the ZIMonSensors.cfg file in /opt/IBM/zimon on all the nodes in the cluster.
5. Start the pmswiftd.service using the following command:

```
systemctl start pmswiftd.service
```

6. Start the pmsensors.service using the following command:

```
systemctl start pmsensors.service
```

Manually upgrading the Performance Monitoring tool

You can upgrade by uninstalling and then reinstalling the new version of the tool.

Prerequisites:

1. Before uninstalling, ensure that you stop sensors and the collector.
2. See "Uninstalling the Performance Monitoring tool" on page 198, for information on uninstallation.

3. See "Manually installing the Performance Monitoring tool" on page 139, for information on manual installation.

To upgrade the tool:

1. Uninstall the tool and then reinstall the new version of the tool.
2. Restart the sensors on the protocol nodes.
3. Restart the collector on the node that previously ran the collector.
4. Verify that the collector and the sensor are running on the node by issuing the **systemctl status pmcollector** command or the **systemctl status pmsensors** command.

or by using the operating system's native package upgrade mechanism. For example, `rpm -Uvh gpfs.gss.pmsensors-version-release.el7.x86_64.rpm`

For more information about the Performance Monitoring tool, see *Configuring the Performance Monitoring tool* in *IBM Spectrum Scale: Advanced Administration Guide*.

Manually installing IBM Spectrum Scale management GUI

The management GUI provides an easy way for the users to configure, manage, and monitor the IBM Spectrum Scale system.

You can install the management GUI by using the following methods:

- Installing management GUI by using the installation GUI. For more information on how to install the management GUI by using the installation GUI, see "Installing IBM Spectrum Scale by using the graphical user interface (GUI)" on page 108.
- Installing management GUI by using installation toolkit. For more information on installing the management GUI by using the installation toolkit, see "Installing IBM Spectrum Scale management GUI by using **spectrumscale** installation toolkit" on page 131.
- Manual installation of the management GUI. The following sections provides the details of how to manually install the management GUI.

Prerequisites

The prerequisites that are applicable for installing the IBM Spectrum Scale system through CLI is applicable for installation through GUI as well. For more information on the prerequisites for installation, see "Installation prerequisites" on page 109.

The Installation rpm that is part of the IBM Spectrum Scale GUI package is necessary for the installation. You need to extract this package to start the installation. The performance tool rpms are also required to enable the performance monitoring tool that is integrated into the GUI. The following rpms are required for performance monitoring tools in GUI:

- The performance tool collector rpm. This rpm is placed only on the collector nodes.
- The performance tool sensor rpm. This rpm is applicable for the sensor nodes, if not already installed.

The following table lists the IBM Spectrum Scale GUI and performance tool package that are required for different platforms.

Table 9. GUI packages required for each platform.

Platform	Package name
RHEL 7.x x86	`gpfs.gui-4.2.0-X.el7.x86_64.rpm` (X is the counter for the latest version starting with 1)
RHEL 7.x ppc64 (big endian)	`gpfs.gui-4.2.0-X.el7.ppc64.rpm` (X is the counter for the latest version starting with 1)

Table 9. GUI packages required for each platform (continued).

Platform	Package name
RHEL 7.x ppc64le (little endian)	`gpfs.gui-4.2.0-X.el7.ppc64le.rpm` (X is the counter for the latest version starting with 1)
SLES12 x86	`gpfs.gui-4.2.0-X.sles12.x86_64.rpm` (X is the counter for the latest version starting with 1)
SLES12 ppc64le (little endian)	`gpfs.gui-4.2.0-X.sles12.ppc64le.rpm` (X is the counter for the latest version starting with 1)
RHEL 6.x x86	`gpfs.gss.pmsensors-4.2.0-0.el6.x86_64gpfs.gss.pmcollector-4.2.0-0.el6.x86_64.rpm`
RHEL 7.x X86	`gpfs.gss.pmcollector-4.2.0-0.el7.x86_64.rpm` `gpfs.gss.pmsensors-4.2.0-0.el7.x86_64.rpm`
RHEL 6.x ppc64	`gpfs.gss.pmcollector-4.2.0-0.el6.ppc64.rpmgpfs.gss.pmsensors-4.2.0-0.el6.ppc64`
RHEL 7.x ppc64	`gpfs.gss.pmcollector-4.2.0-0.el7.ppc64.rpm` `gpfs.gss.pmsensors-4.2.0-0.el7.ppc64.rpm`
RHEL 7.x ppc64 LE	`gpfs.gss.pmcollector-4.2.0-0.el7.ppc64le.rpm` `gpfs.gss.pmsensors-4.2.0-0.el7.ppc64le.rpm`
SLES12 X86	`gpfs.gss.pmcollector-4.2.0-0.SLES12.x86_64.rpm` `gpfs.gss.pmsensors-4.2.0-0.SLES12.X86_64.rpm`
SLES12 ppc64	`gpfs.gss.pmcollector-4.2.0-0.SLES12.ppc64.rpm` `gpfs.gss.pmsensors-4.2.0-0.SLES12.ppc64.rpm`
SLES12 ppc64 LE	`gpfs.gss.pmcollector-4.2.0-0.SLES12.ppc64le.rpm` `gpfs.gss.pmsensors-4.2.0-0.SLES12.ppc64le.rpm`
SLES11 ppc64 (sensor only)	`gpfs.gss.pmsensors-4.2.0-0.SLES11.ppc64.rpm`

Debian sensor packages	`gpfs.gss.pmsensors_4.2.0-0.U14.04_amd64.deb`
	`gpfs.gss.pmsensors_4.2.0-0.U12.04_amd64.deb`
	`gpfs.gss.pmsensors_4.2.0-0.D7.6_amd64.deb`
	`gpfs.gss.pmsensors_4.2.0-0.D6.0.10_amd64.deb`

Ensure that the performance tool collector runs on the same node as the GUI.

Yum repository setup

You can use yum repository to manually install the GUI rpm files. This is the preferred way of GUI installation as yum checks the dependencies and automatically installs missing platform dependencies like the postgres module, which is required but not included in the package.

Installation steps

You can install the management GUI either using the package manager (yum or zypper commands) or by issuing the rpms individually.

Installing management GUI by using package manager (yum or zypper commands)

It is recommended to use this method as the package manager checks the dependencies and automatically installs missing platform dependencies. Issue the following commands to install management GUI:

Red Hat Enterprise Linux

```
yum install gpfs.gss.pmsensors-4.2.0-0.el7.<arch>.rpm
yum install gpfs.gss.pmcollector-4.2.0-0.el7.<arch>.rpm
yum install gpfs.gui-4.2.0-0.el7.<arch>.rpm
```

SLES

```
zypper install gpfs.gss.pmsensors-4.2.0-0.SLES12.<arch>.rpm
zypper install gpfs.gss.pmcollector-4.2.0-0.SLES12.<arch>.rpm
zypper install gpfs.gui-4.2.0-0.sles12.<arch>.rpm
```

Installing management GUI by using rpms

Issue the following commands:

Red Hat Enterprise Linux

```
rpm -ivh gpfs.gss.pmsensors-4.2.0-0.el7.<arch>.rpm
rpm -ivh gpfs.gss.pmcollector-4.2.0-0.el7.<arch>.rpm
rpm -ivh gpfs.gui-4.2.0-0.el7.<arch>.rpm
```

SLES

```
rpm -ivh gpfs.gss.pmsensors-4.2.0-0.SLES12.<arch>.rpm
rpm -ivh gpfs.gss.pmcollector-4.2.0-0.SLES12.<arch>.rpm
rpm -ivh gpfs.gui-4.2.0-0-sles12.<arch>.rpm
```

The sensor rpm must be installed on any additional node that you want to monitor. All sensors must point to the collector node.

Note: The default user name and password to access the IBM Spectrum Scale management GUI is `admin` and `admin001` respectively.

Enabling performance tools in management GUI

The performance tool is installed into `/opt/IBM/zimon`. The following important configuration files are available in this folder:

ZIMonSensors.cfg
: This is the sensor configuration file and it controls which sensors are activated and also sets the reporting interval of each sensor. By setting the reporting interval to -1, a sensor is disabled. A positive number defines the reporting period in seconds. The smallest possible period is once per second.

ZIMonCollector.cfg
: This is the collector configuration file and it defines the number of aggregation levels and the maximum amount of memory that is used. By default, three domains are created: a raw domain that stores the metrics uncompressed, a first aggregation domain that aggregates data to 1-minute averages, and a second aggregation domain that stores data in 15-minute averages. Each domain can be configured with the amount of memory that is used by the in-memory database and also the maximum file size and number of files that are used to store the data on disk.

The startup script of the sensor defines a list of collectors to which data is being sent. By default, the sensor unit reports to a collector that runs on *localhost*. If not, change the sensor configuration to point to the collector IP address.

To enable and initialize the performance tool in the management GUI, do the following:

1. To initialize the performance tool, issue the **systemctl start** command as shown in the following example:

   On collector nodes: `systemctl start pmcollector`

   On all sensor nodes: `systemctl start pmsensors`

   If the performance tool is not configured on your cluster, the system displays the following error messages when you try to start *pmsensors* on the sensor nodes:

   ```
   Job for pmsensors.service failed. See "systemctl status pmsensors.service" and "journalctl -xn" for
   details.
   ```

   To resolve this problem, first configure the cluster for the performance tool by using the **mmperfmon** command. You also need to configure a set of collector nodes while issuing the command as shown in the following example:

   ```
   mmperfmon config generate --collectors [ipaddress/hostname of node1, ipaddress/hostname of node2, ...]
   ```

2. Enable the sensors on the cluster by using the **mmchnode** command. Issuing this command configures and starts the performance tool sensors on the nodes.

   Before issuing the **mmchnode** command, ensure that the *pmsensors* is already installed on all sensor nodes as given in the following example:

   ```
   mmchnode --perfmon -N [SENSOR_NODE_LIST]
   ```

   `[SENSOR_NODE_LIST]` is a comma-separated list of sensor nodes' host names or IP addresses.

   You can also manually configure the performance tools sensor nodes by editing the following file on all sensor nodes: `/opt/IBM/zimon/ZIMonSensors.cfg` . Add the host name or IP address of the node that hosts the collector in the following section for the configuration file:

   ```
   collectors = {
   host = "[HOSTNAME or IP ADDRESS]"
   port = "4739"
   }
   ```

   This specifies the collector to which the sensor is reporting.

3. To show the file system capacity, update the *GPFSDiskCap* file to set frequency in which the capacity needs to be refreshed. You need to specify this value in seconds as shown in the following example:

   ```
   mmperfmon config update GPFSDiskCap.restrict=gui_node GPFSDiskCap.period=86400
   ```

   This sensor must be enabled only on a single node, preferably the GUI node. If this sensor is disabled, the GUI does not show any capacity data. The recommended period is 86400 which means once per day. Since this sensor runs **mmdf**, it is not recommended to use a value less than 10800 (every three hours) for `GPFSDiskCap.period`.

4. Enable quota in the file system to get capacity data on filesets in the GUI. For information on enabling quota, see the **mmchfs -q** option in **mmchfs command** and **mmcheckquota command** in *IBM Spectrum Scale: Administration and Programming Reference*.

5. Start the sensor on every sensor node as shown in the following example:

   ```
   systemctl start pmsensors
   ```

6. After configuring the performance tool, you can start the IBM Spectrum Scale management GUI as shown in the following example:

   ```
   systemctl start gpfsgui
   ```

7. To make sure that the GUI and performance tool are started on the boot process, issue the following commands:

   ```
   systemctl enable gpfsgui.service
   systemctl enable pmsensor.service
   systemctl enable pmcollector.service
   ```

Note: The **pmsensors** and **pmcollector** scripts are **SysV** scripts. On systems that use **systemd** scripts, **systemd** redirects these scripts to **chkconfig**, and the following message will be displayed on the terminal:

```
pmcollector.service is not a native service, redirecting to /sbin/chkconfig.
Executing /sbin/chkconfig pmcollector on the unit files have no [Install] section. They are not meant to be enabled
using systemctl.
Possible reasons for having this kind of units are:
1) A unit may be statically enabled by being symlinked from another unit's
.wants/ or .requires/ directory.
2) A unit's purpose may be to act as a helper for some other unit which has
a requirement dependency on it.
3) A unit may be started when needed via activation (socket, path, timer,
D-Bus, udev, scripted systemctl call, ...).
```

This is not an error message. It is used for information purpose only.

Checking GUI and performance tool status

Issue the **systemctl status gpfsgui** command to know the GUI status as shown in the following example:

```
systemctl status gpfsgui.service
gpfsgui.service - IBM_GPFS_GUI Administration GUI
Loaded: loaded (/usr/lib/systemd/system/gpfsgui.service; disabled)
Active: active (running) since Fri 2015-04-17 09:50:03 CEST; 2h 37min ago
Process: 28141 ExecStopPost=/usr/lpp/mmfs/gui/bin/cfgmantraclient unregister (code=exited, s
tatus=0/SUCCESS)
Process: 29120 ExecStartPre=/usr/lpp/mmfs/gui/bin/check4pgsql (code=exited, status=0/SUCCESS)
Main PID: 29148 (java)
Status: "GSS/GPFS GUI started"
CGroup: /system.slice/gpfsgui.service
≪—29148 /opt/ibm/wlp/java/jre/bin/java -XX:MaxPermSize=256m -Dcom.ibm.gpfs.platform=GPFS
-Dcom.ibm.gpfs.vendor=IBM -Djava.library.path=/opt/ibm/wlp/usr/servers/gpfsgui/lib/
-javaagent:/opt/ibm/wlp/bin/tools/ws-javaagent.jar -jar /opt/ibm/wlp/bin/tools/ws-server.jar gpfsgui
--clean

Apr 17 09:50:03 server-21.localnet.com java[29148]: Available memory in the JVM: 484MB
Apr 17 09:50:03 server.localnet.com java[29148]: Max memory that the JVM will attempt to use: 512MB
Apr 17 09:50:03 server.localnet.com java[29148]: Number of processors available to JVM: 2
Apr 17 09:50:03 server.localnet.com java[29148]: Backend started.
Apr 17 09:50:03 server.localnet.com java[29148]: CLI started.
Apr 17 09:50:03 server.localnet.com java[29148]: Context initialized.
Apr 17 09:50:03 server.localnet.com systemd[1]: Started IBM_GPFS_GUI Administration GUI.
Apr 17 09:50:04 server.localnet.com java[29148]: [AUDIT ] CWWKZ0001I: Application /
started in 6.459 seconds.
Apr 17 09:50:04 server.localnet.com java[29148]: [AUDIT ] CWWKF0012I: The server
installed the following features: [jdbc-4.0, ssl-1.0, localConnector-1.0, appSecurity-2.0,
jsp-2.2, servlet-3.0, jndi-1.0, usr:FsccUserRepo, distributedMap-1.0].
Apr 17 09:50:04 server-21.localnet.com java[29148]: [AUDIT ] CWWKF0011I: ==> When you see
the service was started anything should be OK !
```

Issue the **systemctl status pmcollector** and **systemctl status pmsensors** commands to know the status of the performance tool.

You can also check whether the performance tool backend can receive data by using the GUI or alternative by using a command line performance tool that is called *zc*, which is available in /opt/IBM/zimon folder. For example:

```
echo "get metrics mem_active, cpu_idle, gpfs_ns_read_ops last 10 bucket_size 1" | ./zc 127.0.0.1
Result example:
1: server-21.localnet.com|Memory|mem_active
2: server-22.localnet.com|Memory|mem_active
3: server-23.localnet.com|Memory|mem_active
4: server-21.localnet.com|CPU|cpu_idle
5: server-22.localnet.com|CPU|cpu_idle
6: server-23.localnet.com|CPU|cpu_idle
```

```
7: server-21.localnet.com|GPFSNode|gpfs_ns_read_ops
8: server-22.localnet.com|GPFSNode|gpfs_ns_read_ops
9: server-23.localnet.com|GPFSNode|gpfs_ns_read_ops
Row Timestamp mem_active mem_active mem_active cpu_idle cpu_idle cpu_idle gpfs_ns_read_ops
gpfs_ns_read_ops gpfs_ns_read_ops
1 2015-05-20 18:16:33 756424 686420 382672 99.000000 100.000000 95.980000 0 0 0
2 2015-05-20 18:16:34 756424 686420 382672 100.000000 100.000000 99.500000 0 0 0
3 2015-05-20 18:16:35 756424 686420 382672 100.000000 99.500000 100.000000 0 0 6
4 2015-05-20 18:16:36 756424 686420 382672 99.500000 100.000000 100.000000 0 0 0
5 2015-05-20 18:16:37 756424 686520 382672 100.000000 98.510000 100.000000 0 0 0
6 2015-05-20 18:16:38 774456 686448 384684 73.000000 100.000000 96.520000 0 0 0
7 2015-05-20 18:16:39 784092 686420 382888 86.360000 100.000000 52.760000 0 0 0
8 2015-05-20 18:16:40 786004 697712 382688 46.000000 52.760000 100.000000 0 0 0
9 2015-05-20 18:16:41 756632 686560 382688 57.580000 69.000000 100.000000 0 0 0
10 2015-05-20 18:16:42 756460 686436 382688 99.500000 100.000000 100.000000 0 0 0
```

Node classes used for the management GUI

The IBM Spectrum Scale management GUI automatically creates the following node classes during installation:

- `GUI_SERVERS`: Contains all nodes with a server license and all the GUI nodes
- `GUI_MGMT_SERVERS`: Contains all GUI nodes

Each node on which the GUI services are started is added to these node classes.

For information about removing nodes from these node classes, see "Removing nodes from management GUI-related node class" on page 198.

For information about node classes, see *Specifying nodes as input to GPFS commands* in *IBM Spectrum Scale: Administration and Programming Reference*.

Related concepts:

"Uninstalling the IBM Spectrum Scale management GUI" on page 198
Do the following to uninstall management GUI and remove the performance monitoring components that are installed for the GUI:

"Introduction to IBM Spectrum Scale GUI" on page 36
The IBM Spectrum Scale management GUI provides an easy way to configure and manage various features that are available with the IBM Spectrum Scale system.

Building the GPFS portability layer on Linux nodes

Before starting GPFS, you must build and install the GPFS portability layer.

The GPFS portability layer is a loadable kernel module that allows the GPFS daemon to interact with the operating system.

Note: The GPFS kernel module should be updated any time the Linux kernel is updated. Updating the GPFS kernel module after a Linux kernel update requires rebuilding and installing a new version of the module.

To build the GPFS portability layer on Linux nodes, do the following:

1. Check for the following before building the portability layer:
   - Updates to the portability layer at the IBM Support Portal: Downloads for General Parallel File System (www.ibm.com/support/entry/portal/Downloads/Software/Cluster_software/General_Parallel_File_System).
   - The latest kernel levels supported in the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
2. Build your GPFS portability layer in one of the following ways:

- Using the **mmbuildgpl** command (recommended) for nodes running GPFS 4.1.0.4 or later. For more information, see *mmbuildgpl command* in *IBM Spectrum Scale: Administration and Programming Reference*.
- Using the **Autoconfig** tool.
- Using the directions in **/usr/lpp/mmfs/src/README**.

Using the mmbuildgpl command to build the GPFS portability layer on Linux nodes

Starting with GPFS 4.1.0.4, you can use the **mmbuildgpl** command to simplify the build process.

To build the GPFS portability layer using **mmbuildgpl**, enter the following command:

```
/usr/lpp/mmfs/bin/mmbuildgpl
```

Each kernel module is specific to a Linux version and platform. If you have multiple nodes running exactly the same operating system level on the same platform, and only some of these nodes have a compiler available, you can build the kernel module on one node, then create an installable package that contains the binary module for ease of distribution.

If you choose to generate an installable package for portability layer binaries, perform the following additional step:

```
/usr/lpp/mmfs/bin/mmbuildgpl --build-package
```

When the command finishes, it displays the location of the generated package:

```
Wrote:  /root/rpmbuild/RPMS/x86_64/gpfs.gplbin-2.6.32-279.el6.x86_64-4.2.0-0.x86_64.rpm
```

or

```
Wrote: /tmp/deb/gpfs.gplbin_4.2.0.0_amd64.deb
```

You can then copy the generated package to other machines for deployment. The generated package can *only* be deployed to machines with identical architecture, distribution level, Linux kernel, and IBM Spectrum Scale maintenance level.

Note: During the package generation, temporary files are written to the **/tmp/rpm** or **/tmp/deb** directory, so be sure there is sufficient space available. By default, the generated package goes to **/usr/src/packages/RPMS/<arch>** for SUSE Linux Enterprise Server, **/usr/src/redhat/RPMS/<arch>** for Red Hat Enterprise Linux, and **/tmp/deb** for Debian and Ubuntu Linux.

Using the Autoconfig tool to build the GPFS portability layer on Linux nodes

To simplify the build process, GPFS provides an automatic configuration tool called **Autoconfig**.

The following example shows the commands required to build the GPFS portability layer using the **Autoconfig** tool:

```
cd /usr/lpp/mmfs/src
make Autoconfig
make World
make InstallImages
```

Each kernel module is specific to a Linux version and platform. If you have multiple nodes running exactly the same operating system level on the same platform, or if you have a compiler available on only one node, you can build the kernel module on one node, then create an installable package that contains the binary module for ease of distribution.

If you choose to generate an installable package for portability layer binaries, perform the following additional step:

make rpm for SLES and RHEL Linux
make deb for Debian and Ubuntu Linux

When the command finishes, it displays the location of the generated package:

```
Wrote:  /root/rpmbuild/RPMS/x86_64/gpfs.gplbin-2.6.32-279.el6.x86_64-4.2.0-0.x86_64.rpm
```

or

```
Wrote: /tmp/deb/gpfs.gplbin_4.2.0.0_amd64.deb
```

You can then copy the generated package to other machines for deployment. The generated package can *only* be deployed to machines with identical architecture, distribution level, Linux kernel, and IBM Spectrum Scale maintenance level.

Note: During the package generation, temporary files are written to the **/tmp/rpm** or **/tmp/deb** directory, so be sure there is sufficient space available. By default, the generated package goes to **/usr/src/packages/RPMS/<arch>** for SUSE Linux Enterprise Server, **/usr/src/redhat/RPMS/<arch>** for Red Hat Enterprise Linux, and **/tmp/deb** for Debian and Ubuntu Linux.

For Linux on z Systems: Changing the kernel settings

In order for GPFS to run on Linux on z Systems, the kernel settings need to be changed.

Before starting GPFS, perform the following steps on each Linux on z Systems node.

On SLES 12:

1. In the `/etc/default/grub` file, add the following:

   ```
   GRUB_CMDLINE_LINUX_DEFAULT=" hvc_iucv=8 TERM=dumb osamedium=eth instmode=ftp x
   crashkernel=206M-:103M cio_ignore=all,!ipldev,!condev vmalloc=4096G "
   ```

2. Run the following command:

   ```
   grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

3. Reboot the node.

On SLES11 and Red Hat Enterprise Linux:

1. In the `/etc/zipl.conf` file, add `vmalloc=4096G user_mode=home` as shown in the following example:

   ```
   (10:25:41) dvtc1a:~ # cat /etc/zipl.conf
   # Modified by YaST2. Last modification on Mon May 19 09:39:04 EDT 2014
   [defaultboot]
   defaultmenu = menu

   ###Don't change this comment - YaST2 identifier: Original name: linux###
   [SLES11_SP3_2]
       image = /boot/image-3.0.101-0.15-default
       target = /boot/zipl
       ramdisk = /boot/initrd-3.0.101-0.15-default,0x2000000
       parameters = "root=/dev/mapper/mpatha_part2 hvc_iucv=8 TERM=dumb
    resume=/dev/mapper/mpatha_part1 crashkernel=258M-:129M vmalloc=4096G
   user_mode=home"
   ```

   Note: For SUSE Linux Enterprise Server (SLES) 11 and Red Hat Enterprise Linux 6, `user_mode=home` is optional. For Red Hat Enterprise Linux 7.1 and later releases, this parameter is not required.

2. Run the **zipl** command.

   Note: For information about the **zipl** command, see the topic about the initial program loader for z Systems (-zipl) in Device Drivers, Features, and Commands (www.ibm.com/support/knowledgecenter/api/content/linuxonibm/liaaf/lnz_r_dd.html) in the Linux on z Systems library overview.

3. Reboot the node.

Note: For more detailed information about installation and startup of GPFS on z Systems, see the "Getting started with Elastic Storage for Linux on z Systems based on GPFS technology" white paper, available on the Welcome Page for IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html).

Chapter 4. Installing IBM Spectrum Scale on AIX nodes

There are three steps to installing GPFS on AIX nodes.

Before you begin installation, read "Planning for GPFS" on page 39 and the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Do not attempt to install GPFS if you do not have the prerequisites listed in "Hardware requirements" on page 39 and "Software requirements" on page 40.

Ensure that the **PATH** environment variable on each node includes **/usr/lpp/mmfs/bin**.

The installation process includes:

1. "Creating a file to ease the AIX installation process"
2. "Verifying the level of prerequisite software"
3. "Procedure for installing GPFS on AIX nodes" on page 154

Creating a file to ease the AIX installation process

Creation of a file that contains all of the nodes in your GPFS cluster prior to the installation of GPFS, will be useful during the installation process. Using either host names or IP addresses when constructing the file will allow you to use this information when creating your cluster through the **mmcrcluster** command.

For example, create the file **/tmp/gpfs.allnodes**, listing the nodes one per line:

```
k145n01.dpd.ibm.com
k145n02.dpd.ibm.com
k145n03.dpd.ibm.com
k145n04.dpd.ibm.com
k145n05.dpd.ibm.com
k145n06.dpd.ibm.com
k145n07.dpd.ibm.com
k145n08.dpd.ibm.com
```

Verifying the level of prerequisite software

Before you can install GPFS, verify that your system has the correct software levels installed.

If your system *does not* have the prerequisite AIX level, refer to the appropriate installation manual before proceeding with your GPFS installation. See the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) for the latest software levels.

To verify the software version, run the command:

```
WCOLL=/tmp/gpfs.allnodes dsh "oslevel"
```

The system should display output similar to:

```
7.1.0.0
```

© Copyright IBM Corp. 2014, 2016

Procedure for installing GPFS on AIX nodes

These installation procedures are generalized for all levels of GPFS. Ensure you substitute the correct numeric value for the modification (*m*) and fix (*f*) levels, where applicable. The modification and fix level are dependent upon the current level of program support.

Follow these steps to install the GPFS software using the **installp** command:

1. "Accepting the electronic license agreement"
2. "Creating the GPFS directory"
3. "Creating the GPFS installation table of contents file" on page 155
4. "Installing the GPFS man pages" on page 155
5. "Installing GPFS over a network" on page 155
6. "Verifying the GPFS installation" on page 155

Accepting the electronic license agreement

The GPFS software license agreement is shipped and viewable electronically. The electronic license agreement must be accepted before software installation can continue.

GPFS has three different editions based on functional levels:

- IBM Spectrum Scale Express Edition
- IBM Spectrum Scale Standard Edition
- IBM Spectrum Scale Advanced Edition

For IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition installations, the installation cannot occur unless the appropriate license agreements are accepted. These license agreements are in addition to the license agreement for the IBM Spectrum Scale Express Edition.

When using the **installp** command, use the **-Y** flag to accept licenses and the **-E** flag to view license agreement files on the media.

GPFS license agreements are retained on the AIX system after installation completes. These license agreements can be viewed after installation in the following directories:

`/usr/swlag/GPFS_express` (IBM Spectrum Scale Express Edition)

`/usr/swlag/GPFS_standard` (IBM Spectrum Scale Standard Edition)

`/usr/swlag/GPFS_advanced` (IBM Spectrum Scale Advanced Edition)

Creating the GPFS directory

To create the GPFS directory:

1. On any node create a temporary subdirectory where GPFS installation images will be extracted. For example:

   ```
   mkdir /tmp/gpfslpp
   ```

2. Copy the installation images from the CD-ROM to the new directory, by issuing:

   ```
   bffcreate -qvX -t /tmp/gpfslpp -d /dev/cd0/AIX all
   ```

   This command places these GPFS installation files in the images directory:

   `gpfs.base`

   `gpfs.docs.data`

   `gpfs.msg.en_US`

   `gpfs.gskit`

   `gpfs.ext` (IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition only)

`gpfs.crypto` (IBM Spectrum Scale Advanced Edition only)

`gpfs.adv` (IBM Spectrum Scale Advanced Edition only)

3. If GPFS Native RAID is not intended to be used, the GPFS Native RAID installation file `gpfs.gnr` can be removed at this point to conserve disk space. For more information about GPFS Native RAID, see *IBM Spectrum Scale RAID: Administration*.

Creating the GPFS installation table of contents file

To create the GPFS installation table of contents file:

1. Make the new image directory the current directory:
   ```
   cd /tmp/gpfslpp
   ```
2. Use the **inutoc .** command to create a **.toc** file. The **.toc** file is used by the **installp** command.
   ```
   inutoc .
   ```

Installing the GPFS man pages

In order to use the GPFS man pages you must install the **gpfs.docs.data** image.

The GPFS manual pages will be located at **/usr/share/man/**.

Installation consideration: The **gpfs.docs.data** image need not be installed on all nodes if man pages are not desired or local file system space on the node is minimal.

Installing GPFS over a network

Install GPFS according to these directions, where *localNode* is the name of the node on which you are running:

1. If you are installing on a shared file system network, ensure the directory where the GPFS images can be found is NFS exported to all of the nodes planned for your GPFS cluster (**/tmp/gpfs.allnodes**).
2. Ensure an acceptable directory or mountpoint is available on each target node, such as **/tmp/gpfslpp**. If there is not, create one:
   ```
   WCOLL=/tmp/gpfs.allnodes dsh "mkdir /tmp/gpfslpp"
   ```
3. If you are installing on a shared file system network, to place the GPFS images on each node in your network, issue:
   ```
   WCOLL=/tmp/gpfs.allnodes dsh "mount localNode:/tmp/gpfslpp /tmp/gpfslpp"
   ```

   Otherwise, issue:
   ```
   WCOLL=/tmp/gpfs.allnodes dsh "rcp localNode:/tmp/gpfslpp/gpfs* /tmp/gpfslpp"
   WCOLL=/tmp/gpfs.allnodes dsh "rcp localNode:/tmp/gpfslpp/.toc /tmp/gpfslpp"
   ```
4. Install GPFS on each node:
   ```
   WCOLL=/tmp/gpfs.allnodes dsh "installp -agXYd /tmp/gpfslpp gpfs"
   ```

Verifying the GPFS installation

Verify that the installation procedure placed the required GPFS files on each node by running the **lslpp** command on *each* node:

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

The system should return output similar to the following:

```
  Fileset             4.2.0-0         Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  gpfs.adv                   4.2.0-0  COMMITTED  GPFS File Manager
  gpfs.base                  4.2.0-0  COMMITTED  GPFS File Manager
  gpfs.crypto                4.2.0-0  COMMITTED  GPFS Cryptographic Subsystem
  gpfs.ext                   4.2.0-0  COMMITTED  GPFS Extended Features
```

```
  gpfs.gskit               8.0.50.47  COMMITTED  GPFS GSKit Cryptography
                                                 Runtime
  gpfs.msg.en_US             4.2.0-0 COMMITTED  GPFS Server Messages - U.S.
                                                 English

Path: /etc/objrepos
  gpfs.base                  4.2.0-0  COMMITTED  GPFS File Manager

Path: /usr/share/lib/objrepos
  gpfs.docs.data             4.2.0-0  COMMITTED  GPFS Server Man Pages and
                                                 Documentation
```

The output that is returned on your system can vary depending on whether you have IBM Spectrum Scale Express Edition, IBM Spectrum Scale Standard Edition or, IBM Spectrum Scale Advanced Edition installed.

Note: The path returned by `lslpp -l` shows the location of the package control data used by **installp**. The listed path does not show GPFS file locations. To view GPFS file locations, use the `-f` flag.

Chapter 5. Installing IBM Spectrum Scale on Windows nodes

There are several steps to installing GPFS on Windows nodes. The information in this topic will point you to the detailed steps.

Before you begin installation, read the following:

- "Planning for GPFS" on page 39
- The IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html)
- "GPFS for Windows overview" and all of its subtopics

Do not install GPFS unless you have the prerequisites listed in "Hardware requirements" on page 39 and "Software requirements" on page 40.

The installation process includes:

1. "Installing GPFS prerequisites" on page 161
2. "Procedure for installing GPFS on Windows nodes" on page 164
3. "Configuring a mixed Windows and UNIX cluster" on page 166

To install GPFS for Windows, first configure your Windows systems as described in "Installing GPFS prerequisites" on page 161. This includes steps such as joining an Active Directory domain and installing Cygwin from the Cygwin website (www.cygwin.com), which provides a Unix-like environment on Windows. GPFS installation is simple once the prerequisites are completed. Finally, if your system will be part of a cluster that includes UNIX nodes, follow the steps described in "Configuring a mixed Windows and UNIX cluster" on page 166. This includes creating the GPFS Administration service, installing OpenSSH, and other requirements. Complete this process before performing configuration steps common to all GPFS supported platforms.

Note: Throughout this information, UNIX file name conventions are used. For example, the GPFS cluster configuration data is stored in the **/var/mmfs/gen/mmsdrfs** file. On Windows, the UNIX namespace starts under the Cygwin installation directory, which by default is **%SystemDrive%\cygwin64**, so the GPFS cluster configuration data is stored in the **C:\cygwin64\var\mmfs\gen\mmsdrfs** file.

GPFS for Windows overview

GPFS for Windows participates in a new or existing GPFS cluster in conjunction with AIX and Linux systems.

Support includes the following:

- Core GPFS parallel data services
- Windows file system semantics
- A broad complement of advanced GPFS features
- User identity mapping between Windows and UNIX
- Access to GPFS 3.4 and/or later file systems

Identity mapping between Windows and UNIX user accounts is a key feature. System administrators can explicitly match users and groups defined on UNIX with those defined on Windows. Users can maintain file ownership and access rights from either platform. System administrators are not required to define an identity map. GPFS automatically creates a mapping when one is not defined.

© Copyright IBM Corp. 2014, 2016

GPFS supports the unique semantic requirements posed by Windows. These requirements include case-insensitive names, NTFS-like file attributes, and Windows file locking. GPFS provides a bridge between a Windows and POSIX view of files, while not adversely affecting the functions provided on AIX and Linux.

GPFS for Windows provides the same core services to parallel and serial applications as are available on AIX and Linux. GPFS gives parallel applications simultaneous access to files from any node that has GPFS mounted, while managing a high level of control over all file system operations. System administrators and users have a consistent command interface on AIX, Linux, and Windows. With few exceptions, the commands supported on Windows are identical to those on other GPFS platforms. See "GPFS limitations on Windows" for a list of commands that Windows clients do not support.

GPFS support for Windows

The IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) lists the levels of the Windows operating system that are supported by GPFS.

Limited GPFS support for the Windows platform first appeared in GPFS 3.2.1. Subsequent GPFS releases have expanded the set of available functions and features so that now most GPFS capabilities supported on the UNIX platforms are also supported on Windows.

GPFS limitations on Windows

GPFS for Windows does not fully support all of the GPFS features that are available on AIX and Linux. Some of these limitations constrain how you can configure a GPFS cluster when it includes Windows nodes. The remaining limitations only pertain to Windows nodes rather than the whole cluster.

GPFS for Windows imposes some constraints on how you can configure and operate a cluster when it includes Windows nodes. File systems must be created with GPFS 3.2.1.5 or higher. Windows nodes can only mount file systems that were formatted with GPFS versions starting with 3.2.1.5. There is no support for upgrading existing file systems that were created with GPFS 3.1 or earlier.

The remaining GPFS for Windows limitations only pertain to the Windows nodes in a cluster:

- The following GPFS commands are not supported on Windows:
  - **mmafmctl**
  - **mmafmconfig**
  - **mmafmlocal**
  - **mmapplypolicy**
  - **mmbackup**
  - **mmbackupconfig**, **mmrestoreconfig**
  - **mmcheckquota**, **mmdefedquota**, **mmedquota**, **mmlsquota**, **mmrepquota**
  - **mmclone**
  - **mmdelacl**, **mmeditacl**, **mmgetacl**, **mmputacl**
  - **mmimgbackup**, **mmimgrestore**
  - **mmpmon**
- The GPFS Application Programming Interfaces (APIs) are not supported on Windows.
- The native Windows backup utility is not supported.
- Symbolic links that are created on UNIX-based nodes are specially handled by GPFS Windows nodes; they appear as regular files with a size of 0 and their contents cannot be accessed or modified.
- GPFS on Windows nodes attempts to preserve data integrity between memory-mapped I/O and other forms of I/O on the same computation node. However, if the same file is memory mapped on more than one Windows node, data coherency is not guaranteed between the memory-mapped sections on

these multiple nodes. In other words, GPFS on Windows does not provide distributed shared memory semantics. Therefore, applications that require data coherency between memory-mapped files on more than one node might not function as expected.
- GPFS on Windows is not supported on the AFM cluster.

File system name considerations

GPFS file system names should be no longer than 31 characters if the file system will be mounted on a Windows node. GPFS file system names are used as Windows file system labels, and Windows limits the length of these labels to 31 characters. Any attempt to mount a file system with a long name will fail.

You can rename a file system using a command like the following:

```
mmchfs gpfs_file_system_name_that_is_too_long -W gpfs_name_1
```

File name considerations

File names created on UNIX-based GPFS nodes that are not valid on Windows are transformed into valid short names. A name may not be valid either because it is a reserved name like NUL or COM2, or because it has a disallowed character like colon (:) or question mark (?).

See the MSDN Library description of Naming Files, Paths, and Namespaces (msdn.microsoft.com/en-us/library/aa365247(VS.85).aspx) for complete details.

Windows applications can use short names to access GPFS files with Windows file names that are not valid. GPFS generates unique short names using an internal algorithm. You can view these short names using **dir /x** in a DOS command prompt.

Table 10 shows an example:

Table 10. Generating short names for Windows

UNIX	Windows
foo+bar.foobar	FO~23Q_Z.foo
foo|bar.-bar	FO~TD}C5._ba
f|bar.-bary	F_AMJ5!._ba

Case sensitivity

Native GPFS is case-sensitive; however, Windows applications can choose to use case-sensitive or case-insensitive names.

This means that case-sensitive applications, such as those using Windows support for POSIX interfaces, behave as expected. Native Win32 applications (such as Windows Explorer) have only case-aware semantics.

The case specified when a file is created is preserved, but in general, file names are case insensitive. For example, Windows Explorer allows you to create a file named **Hello.c**, but an attempt to create **hello.c** in the same folder will fail because the file already exists. If a Windows node accesses a folder that contains two files that are created on a UNIX node with names that differ only in case, Windows inability to distinguish between the two files might lead to unpredictable results.

Antivirus software

If more than one GPFS Windows node is running antivirus software that scans directories and files, shared files only need to be scanned by one GPFS node. It is not necessary to scan shared files more than once.

When you run antivirus scans from more than one node, schedule the scans to run at different times to allow better performance of each scan, as well as to avoid any conflicts that might arise because of concurrent exclusive access attempts by the antivirus software from multiple nodes. Note that enabling real-time antivirus protection for GPFS volumes could significantly degrade GPFS performance and cause excessive resource consumption.

Tip: Consider using a single, designated Windows node to perform all virus scans.

Differences between GPFS and NTFS

GPFS differs from the Microsoft Windows NT File System (NTFS) in its degree of integration into the Windows administrative environment, Windows Explorer, and the desktop.

The differences are as follows:

- Manual refreshes are required to see any updates to the GPFS namespace.
- You cannot use the recycle bin.
- You cannot use distributed link tracking. This is a technique through which shell shortcuts and OLE links continue to work after the target file is renamed or moved. Distributed link tracking can help you locate the link sources in case the link source is renamed or moved to another folder on the same or different volume on the same computer, or moved to a folder on any computer in the same domain.
- You cannot use NTFS change journaling. This also means that you cannot use the Microsoft Indexing Service or Windows Search Service to index and search files and folders on GPFS file systems.

GPFS does not support the following NTFS features:

- File compression (on individual files or on all files within a folder)
- Encrypted directories
- Encrypted files (GPFS file encryption is administered through GPFS-specific commands. For more information, see *Encryption* in *IBM Spectrum Scale: Advanced Administration Guide*.)
- Quota management (GPFS quotas are administered through GPFS-specific commands)
- Reparse points
- Defragmentation and error-checking tools
- Alternate data streams
- Directory Change Notification
- The assignment of an access control list (ACL) for the entire drive
- A change journal for file activity
- The scanning of all files or directories that a particular SID owns (**FSCTL_FIND_FILES_BY_SID**)
- Generation of AUDIT and ALARM events specified in a System Access Control List (SACL). GPFS is capable of storing SACL content, but will not interpret it.
- Windows sparse files API
- Transactional NTFS (also known as TxF)

Access control on GPFS file systems

GPFS provides support for the Windows access control model for file system objects.

Each GPFS file or directory has a Security Descriptor (SD) object associated with it and you can use the standard Windows interfaces for viewing and changing access permissions and object ownership (for example, Windows Explorer Security dialog panel). Internally, a Windows SD is converted to an NFS V4 access control list (ACL) object, which ensures that access control is performed consistently on other supported operating systems. GPFS supports all discretionary access control list (DACL) operations, including inheritance. GPFS is capable of storing system access control list (SACL) objects, but generation of AUDIT and ALARM events specified in SACL contents is not supported.

An important distinction between GPFS and Microsoft Windows NT File Systems (NTFS) is the default set of permissions for the root (top-level) directory on the file system. On a typical NTFS volume, the DACL for the top-level folder has several inheritable entries that grant full access to certain special accounts, as well as some level of access to nonprivileged users. For example, on a typical NTFS volume, the members of the local group **Users** would be able to create folders and files in the top-level folder. This approach differs substantially from the traditional UNIX convention where the root directory on any file system is only writable by the local **root** superuser by default. GPFS adheres to the latter convention; the root directory on a new file system is only writable by the UNIX user **root**, and does not have an extended ACL when the file system is created. This is to avoid impacting performance in UNIX-only environments, where the use of extended ACLs is not common.

When a new GPFS file system is accessed from a Windows client for the first time, a security descriptor object is created for the root directory automatically, and it will contain a noninheritable DACL that grants full access to the local **Administrators** group and read-only access to the local **Everyone** group. This allows only privileged Windows users to create new files and folders. Because the root directory DACL has no inheritable entries, new objects will be created with a default DACL that grants local **Administrators** and **SYSTEM** accounts full access. Optionally, the local system administrator could create a subdirectory structure for Windows users, and explicitly set DACLs on new directories as appropriate (for example, giving the necessary level of access to nonprivileged users).

Note: Some applications expect to find NTFS-style permissions on all file systems and they might not function properly when that is not the case. Running such an application in a GPFS folder where permissions have been set similar to NTFS defaults might correct this.

Installing GPFS prerequisites

This topic provides details on configuring Windows systems prior to installing GPFS.

Perform the following steps:

1. "Configuring Windows" on page 162
   a. "Assigning a static IP address" on page 162
   b. "Joining an Active Directory domain" on page 162
   c. "Disabling User Account Control" on page 162
   d. "Disabling the Windows firewall" on page 162
   e. "Installing the Tracefmt and Tracelog programs (optional)" on page 163
2. From here on, all GPFS installation steps must be executed as a special user with Administrator privileges. Further, all GPFS administrative operations (such as creating a cluster and file systems) must also be issued as the same user. This special user will depend on your cluster type. For Windows homogeneous clusters, run GPFS commands as a member of the **Domain Admins** group or as a domain account that is a member of the local **Administrators** group. For clusters with both Windows and UNIX nodes, run GPFS commands as **root**, a special domain user account described in "Creating the GPFS administrative account" on page 167.
3. "Installing Cygwin" on page 163

See the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) for the latest:

- Software recommendations
- Configuration information

For additional requirements when setting up clusters that contain both Windows nodes and AIX or Linux nodes, see "Configuring a mixed Windows and UNIX cluster" on page 166.

Configuring Windows

This topic provides some details on installing and configuring Windows on systems that will be added to a GPFS cluster.

The IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html) lists the levels of the Windows operating system that are supported by GPFS.

Assigning a static IP address

GPFS communication requires invariant static IP addresses for each GPFS node.

Joining an Active Directory domain

All Windows systems in the same GPFS cluster should be members of the same Active Directory domain. Join these systems to the Windows domain before adding them to a GPFS cluster.

GPFS expects that all Windows nodes in a cluster are members of the same domain. This gives domain users a consistent identity and consistent file access rights independent of the system they are using. The domain controllers, which run the Active Directory Domain Services, are not required to be members of the GPFS cluster.

Refer to your Windows Server documentation for information on how to install and administer Active Directory Domain Services.

Disabling User Account Control

On Windows Server 2008 nodes, you must disable User Account Control (UAC) for GPFS to operate correctly. UAC needs to be disabled for the entire system, not just turned off for GPFS administrative users.

Windows Server 2008 R2 and Windows Server 2012 (including R2) do not have this requirement. However, if UAC is enabled, some GPFS functions may not work properly. Therefore, it is recommended that UAC be disabled, regardless of the Windows OS version. On the systems with UAC enabled, GPFS administrative users run in **Admin Mode** such as when a program is started with the **Run as Administrator** option.

To disable UAC on Windows systems, follow these steps:

1. Open the **System Configuration** application under **Administrative Tools**.
2. Select the **Tools** tab and scroll down to select **Disable UAC**.
3. Click **Launch**.

Note: This change requires a reboot.

Limitations: Windows is responsible for providing file ownership information to GPFS. If UAC is disabled, the default ownership of files may be affected. Files from users who are members of an administrative group will be owned by the administrative group, but files from users without administrative membership will be mapped to their user ID. This restriction is documented on the Microsoft Support website (support.microsoft.com/kb/947721).

Disabling the Windows firewall

GPFS requires that you modify the default Windows Firewall settings.

The simplest change that will allow GPFS to operate properly is to disable the firewall. Open **Windows Firewall** in the Control Panel and click **Turn Windows firewall on or off**, and select **Off** under the General tab. For related information, see *GPFS port usage* in *IBM Spectrum Scale: Advanced Administration Guide*.

Installing the Tracefmt and Tracelog programs (optional)

GPFS diagnostic tracing (**mmtracectl**) on Windows uses the Microsoft programs called **tracefmt.exe** and **tracelog.exe**. These programs are not included with Windows but can be downloaded from Microsoft. The **tracefmt.exe** and **tracelog.exe** programs are only for tracing support and are not required for normal GPFS operations.

The **tracefmt.exe** and **tracelog.exe** programs are included with the Windows Driver Kit (WDK) as well as the Windows SDK. You can download either of these packages from the Microsoft Download Center (www.microsoft.com/download).

To allow GPFS diagnostic tracing on Windows using the WDK, follow these steps:

1. Download the latest version of the Windows Driver Kit (WDK) from Microsoft.
2. Install the Tools feature of the WDK on some system to obtain a copy of **tracefmt.exe** and **tracelog.exe**.
3. Locate and copy **tracefmt.exe** and **tracelog.exe** to both the **/usr/lpp/mmfs/bin** and **/usr/lpp/mmfs/win** directories to ensure **mmtracectl** properly locates these programs.

For additional information about GPFS diagnostic tracing, see *The GPFS trace facility* in *IBM Spectrum Scale: Problem Determination Guide*.

Installing Cygwin

Cygwin is a POSIX environment available for Windows and can be downloaded from the Cygwin website (www.cygwin.com). GPFS uses this component to support many of its administrative scripts. System administrators have the option of using either the **GPFS Admin Command Prompt** or **GPFS Admin Korn Shell** to run GPFS commands.

Cygwin must be installed before installing GPFS. It is a software package that provides a Unix-like environment on Windows and provides runtime support for POSIX applications and includes programs such as **grep**, **ksh**, **ls**, and **ps**.

When running Cygwin setup, only the standard packages are installed by default. GPFS requires installation of additional packages, which are listed in "Installing the 64-bit version of Cygwin" on page 164.

Note: Starting with GPFS 4.1.1, the 32-bit version of Cygwin is no longer supported for Windows nodes running GPFS. Users that are running GPFS 4.1 with the 32-bit version of Cygwin installed must upgrade to the 64-bit version of Cygwin before installing GPFS 4.1.1. For more information, see "Upgrading from the 32-bit version of Cygwin to the 64-bit version of Cygwin." For users on GPFS releases prior to 4.1 (SUA based), refer to one of the following sections that is appropriate for your installation:

- "Migrating to IBM Spectrum Scale V4.2 from GPFS V3.5" on page 175
- "Migrating to IBM Spectrum Scale 4.2 from GPFS 3.4 or GPFS 3.3" on page 176

Upgrading from the 32-bit version of Cygwin to the 64-bit version of Cygwin

Follow these instructions to upgrade from the 32-bit version of Cygwin to the 64-bit version of Cygwin:

1. Uninstall GPFS 4.1 and reboot.
2. Uninstall IBM GSKit for GPFS and reboot.
3. Uninstall the GPFS 4.1 license.
4. Stop and delete any Cygwin 32-bit services, such as OpenSSH, that might have been configured.
5. Do *not* uninstall the 32-bit version of Cygwin yet, or you may lose GPFS configuration information.
6. Install the 64-bit version of Cygwin using the instructions in "Installing the 64-bit version of Cygwin" on page 164.

7. Install the GPFS 4.1.1 license for the appropriate edition; for example, `gpfs.ext-4.1.1-Windows-license.msi`.
8. Install the appropriate GPFS 4.1.1 edition; for example, `gpfs.ext-4.1.1.x-Windows.msi`.
9. Install IBM GSKit for GPFS.
10. Uninstall the 32-bit version of Cygwin completely.
11. Follow the procedures in "Installing and configuring OpenSSH" on page 168.

Installing the 64-bit version of Cygwin

To install the 64-bit version of Cygwin for Windows, follow these steps:

1. Logon to the Windows node as the account you will use for GPFS administrative operations.
2. Go to the Cygwin website (www.cygwin.com), and click on the **Install Cygwin** link in the upper-left pane.
3. Download and start the installation of the **setup-x86_64.exe** file.
4. Follow the prompts to continue with the install. The default settings are recommended until the Select Packages dialog is displayed. Then, select the following packages (use the Search field to quickly find these packages):
   - diffutils: A GNU collection of diff utilities
   - flip: Convert text file line endings between Unix and DOS formats
   - m4: GNU implementation of the traditional Unix macro processor
   - mksh: MirBSD Korn Shell
   - perl: Perl programming language interpreter
   - procps: System and process monitoring utilities
   - openssh: The OpenSSH server and client programs (only required if you plan on mixing Linux, AIX and Windows nodes in the same cluster)
5. Click the **Next** button, and continue to follow the prompts to complete the installation.
6. Follow the procedures in "Installing and configuring OpenSSH" on page 168.

Procedure for installing GPFS on Windows nodes

IBM provides GPFS as Windows Installer packages (MSI), which allow both interactive and unattended installations. Perform the GPFS installation steps as the **Administrator** or some other member of the **Administrators** group.

Before installing GPFS on Windows nodes, verify that all the installation prerequisites have been met. For more information, see "Installing GPFS prerequisites" on page 161.

To install GPFS on a Windows node, follow these steps:

1. Run *one* (not both) of the following license installation packages from the product media (depending on the GPFS edition you wish to install) and accept the license:

   gpfs.base-4.2.0-Windows-license.msi (IBM Spectrum Scale Express Edition) or **gpfs.ext-4.2.0-Windows-license.msi** (IBM Spectrum Scale Standard Edition)

   Unlike on Linux and AIX, the various IBM Spectrum Scale Editions on Windows are fully self-contained and mutually exclusive. Only one edition can be installed at any given time, and each requires its corresponding license package.

   Rebooting is normally not required.

   Note: All nodes in a cluster must have the same edition installed.
2. Download and install the latest service level of GPFS from the IBM Support Portal: Downloads for General Parallel File System (www.ibm.com/support/entry/portal/Downloads/Software/

Cluster_software/General_Parallel_File_System). The GPFS package name includes the GPFS version (for example, **gpfs.ext-4.2.*x.x*-Windows.msi**). For the latest information on the supported Windows operating system versions, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Under some circumstances, the installation process will prompt you to reboot the systems when it is required. You do not need to install the GPFS 4.2.0.0 package included on the media before installing the latest update.

3. Download and install the latest level of IBM GSKit for GPFS. GSKit (**gpfs.gskit-*x.x.x.x*.msi**) comes as a single package that works with both the IBM Spectrum Scale Express Edition and the IBM Spectrum Scale Standard Edition.

   Note: For this release, the IBM GSKit version must be at least **8.0.50.47** or higher.

To upgrade GPFS, do the following:

1. Uninstall your current GPFS packages (for example, **gpfs.base-4.*x.x*.0-Windows.msi**, **gpfs.ext-4.*x.x*.0-Windows.msi**, and **gpfs.gskit-8.0.50.16.msi**).
2. Reboot.
3. Install the latest PTF packages (for example, **gpfs.ext-4.2.0.*x*-Windows.msi**).

When upgrading, you do not need to uninstall and reinstall the license package unless you are explicitly instructed to do so by IBM. In addition, **gpfs.gskit** only needs to be upgraded if the update (zip file) contains a newer version.

For more information, refer to "Steps to establishing and starting your GPFS cluster" on page 101.

The GPFS installation package provides some options that you can select by installing GPFS from a Command Prompt. Property values control the options. The command line syntax is:

```
msiexec.exe /package <Package> [Optional Parameters] [Property=Value]
```

The following property values are available:

AgreeToLicense=yes
: This property applies to the license packages (**gpfs.base-4.2.0-Windows-license.msi** and **gpfs.ext-4.2.0-Windows-license.msi**). It allows GPFS to support an installation procedure that does not require user input. IBM recommends that you perform at least one interactive installation on a typical system before attempting an unattended installation. This will help identify any system configuration issues that could cause the installation to fail.

  The following command installs GPFS without prompting you for input or generating any error dialog boxes:

  ```
  msiexec /package gpfs.ext-4.2.0-Windows-license.msi /passive AgreeToLicense=yes
  ```

  The **msiexec.exe** executable file supports display options other than **/passive**. See **msiexec** documentation for details.

RemoteShell=no
: This property applies to the product package (for example, **gpfs.ext-4.2.0-Windows-license.msi**). It is equivalent to running **mmwinservctl set –remote-shell no**, but performs this configuration change before **mmwinserv** initially starts. This option is available to satisfy security policies that restrict network communication protocols.

You can verify that GPFS is installed correctly by opening the **Programs and Features** control panel. **IBM Spectrum Scale** should be included in the list of installed programs. The program's version should match the version of the update package.

The GPFS software license agreement is shipped and is viewable electronically. The license agreement will remain available in the **%SystemDrive%\Program Files\IBM\GPFS\4.2.0.0\license** directory for future access.

Running GPFS commands

Once GPFS is installed, the account you use for GPFS administrative operations (such as creating a cluster and file systems) will depend on your cluster type.

For Windows clusters, run GPFS commands as a member of the **Domain Admins** group or as a domain account that is part of the local **Administrators** group. For clusters with both Windows and UNIX nodes, run GPFS commands as **root**, a special domain user account described in "Creating the GPFS administrative account" on page 167.

You must run all GPFS commands from either a **GPFS Admin Command Prompt** or a **GPFS Admin Korn Shell**. The GPFS administrative commands may not work properly if issued from the Cygwin Terminal. Open a new **GPFS Admin Command Prompt** or **GPFS Admin Korn Shell** after installing GPFS so that it uses the updated **PATH** environment variable required to execute GPFS commands.

GPFS Admin Command Prompt and **GPFS Admin Korn Shell** can be accessed using their desktop shortcuts or from under the **IBM GPFS** Start menu group. Each of these program links starts its respective shell using the **Run as Administrator** option.

Configuring a mixed Windows and UNIX cluster

For GPFS clusters that include both Windows and UNIX nodes, this topic describes the additional configuration steps needed beyond those described in "Installing GPFS prerequisites" on page 161.

For mixed clusters, perform the following steps:

1. Optionally, install and configure identity mapping on your Active Directory domain controller (see "Identity Management for UNIX (IMU)").
2. Create the root administrative account (see "Creating the GPFS administrative account" on page 167).
3. Edit the Domain Group Policy to give root the right to log on as a service (see "Allowing the GPFS administrative account to run as a service" on page 167.
4. Configure the GPFS Administration service (**mmwinserv**) to run as root (see "Configuring the GPFS Administration service" on page 167).
5. Install and configure OpenSSH (see "Installing and configuring OpenSSH" on page 168).

Complete this process before performing configuration steps common to all GPFS supported platforms.

Identity Management for UNIX (IMU)

GPFS can exploit a Windows Server feature called Identity Management for UNIX (IMU) to provide consistent identities among all nodes in a cluster.

GPFS expects that all Windows nodes in a cluster are members of the same Active Directory domain. This gives domain users a consistent identity and consistent file access rights independent of the system they are using.

In addition, GPFS can exploit the Identity Management for UNIX (IMU) service for mapping users and groups between Windows and UNIX. IMU is an optional component of Microsoft Windows Server (starting with Server 2003 R2) that can be installed on domain controllers. GPFS does not require IMU.

For IMU installation and configuration information, see *Identity management on Windows* in *IBM Spectrum Scale: Advanced Administration Guide*.

Creating the GPFS administrative account

GPFS uses an administrative account in the Active Directory domain named **root** in order to interoperate with UNIX nodes in the cluster. Create this administrative account as follows:

1. Create a domain user with the logon name **root**.
2. Add user **root** to the **Domain Admins** group or to the local **Administrators** group on each Windows node.
3. In **root Properties/Profile/Home/LocalPath**, define a **HOME** directory such as **C:\Users\root\home** that does not include spaces in the path name and is not the same as the profile path.
4. Give **root** the right to log on as a service as described in "Allowing the GPFS administrative account to run as a service."

Step 3 is required for the Cygwin environment (described in "Installing Cygwin" on page 163) to operate correctly. Avoid using a path that contains a space character in any of the path names. Also avoid using **root**'s profile path (for example, **C:\User\root**). OpenSSH requires specific permissions on this directory, which can interfere with some Windows applications.

You may need to create the **HOME** directory on each node in your GPFS cluster. Make sure that **root** owns this directory.

Allowing the GPFS administrative account to run as a service

Clusters that depend on a **root** account to interoperate with UNIX nodes in a cluster will need to configure the GPFS Administrative Service (**mmwinserv**) to run as the **root** account. For this, **root** needs to be assigned the right to log on as a service. See "Configuring the GPFS Administration service" for details.

The right to log on as a service is controlled by the Local Security Policy of each Windows node. You can use the Domain Group Policy to set the Local Security Policy on all Windows nodes in a GPFS cluster.

The following procedure assigns the *log on as a service* right to an account when the domain controller is running on Windows Server 2008:

1. Open **Group Policy Management** (available under **Administrative Tools**).
2. In the console tree, expand ***Forest name*/Domains/*Domain name*/Group Policy Objects**.
3. Right click **Default Domain Policy** and select **Edit**.
4. In the console tree of the Group Policy Management Editor, expand down to **Computer Configuration/Policies/Windows Settings/Security Settings/Local Policies/User Rights Assignment**.
5. Double click the **Log on as a service policy**.
6. Check **Define these policy settings if necessary**.
7. Use **Add User or Group...** to include the ***DomainName*\root** account in the policy, then click **OK**.

Refer to your *Windows Server* documentation for a full explanation of Local Security Policy and Group Policy Management.

Configuring the GPFS Administration service

GPFS for Windows includes a service called **mmwinserv**. In the Windows Services management console, this service has the name GPFS Administration. **mmwinserv** supports GPFS operations such as **autoload** and remote command execution in Windows GPFS clusters. The Linux and AIX versions of GPFS do not have a similar component. The **mmwinserv** service is used on all Windows nodes starting with GPFS 3.3.

The GPFS installation package configures **mmwinserv** to run using the default **LocalSystem** account. This account supports Windows GPFS clusters. For clusters that include both Windows and UNIX nodes, you

must configure **mmwinserv** to run as **root**, the GPFS administrative account. Unlike **LocalSystem**, **root** can access the Identity Management for UNIX (IMU) service and can access other GPFS nodes as required by some cluster configurations.

For IMU installation and configuration information, see *Identity management on Windows* in *IBM Spectrum Scale: Advanced Administration Guide*. For information on supporting administrative access to GPFS nodes, see the *Requirements for administering a GPFS file system* topic in the *IBM Spectrum Scale: Administration and Programming Reference*.

Before configuring **mmwinserv** to run as **root**, you must first grant **root** the right to run as a service. For details, see "Allowing the GPFS administrative account to run as a service" on page 167.

Use the GPFS command **mmwinservctl** to set and maintain the GPFS Administration service's log on account. **mmwinservctl** must be run on a Windows node. You can run **mmwinservctl** to set the service account before adding Windows nodes to a cluster. You can also use this command to change or update the account on nodes that are already in a cluster. GPFS can be running or stopped when executing **mmwinservctl**, however, refrain from running other GPFS administrative commands at the same time.

In this example, **mmwinservctl** configures three nodes before they are added to a GPFS cluster containing both Windows and UNIX:

```
mmwinservctl set -N node1,node2,node3 --account mydomain/root --password mypwd --remote-shell no
```

Whenever root's password changes, the **mmwinserv** logon information needs to be updated to use the new password. The following command updates on all Windows nodes in a cluster with a new password:

```
mmwinservctl set -N all --password mynewpwd
```

As long as **mmwinserv** is running, the service will not be affected by an expired or changed password and GPFS will continue to function normally. However, GPFS will not start after a system reboot when **mmwinserv** is configured with an invalid password. If for any reason the Windows domain or root password changes, then **mmwinservctl** should be used to update the domain and password. The domain and password can also be updated on a per node basis by choosing **Administrative Tools** > **Computer Management** > **Services and Applications** > **Services**, and selecting **GPFS Administration**. Choose **File** > **Properties** > **Logon** and update the <*domain*>*username* and the password.

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

Installing and configuring OpenSSH

If using a mixed cluster, OpenSSH must be configured on the Windows nodes. Refer to the Cygwin FAQ (www.cygwin.com/faq.html) and documentation on how to setup **sshd**. Replace the usage of the account **cyg_server** in the Cygwin documentation with **root** when setting up a privileged account for **sshd**.

The following are some guidelines in addition to the Cygwin instructions on setting up **sshd**:

1. Verify that all nodes can be pinged among themselves by host name, Fully Qualified Domain Name (FQDN) and IP address.
2. If not using IPv6, disable it. For more information, see How to disable IPv6 or its components in Windows (support.microsoft.com/kb/929852).
3. Check that `passwd` contains the privileged user that you plan to use for GPFS operations, as well as its correct home path:

```
$ cat /etc/passwd | grep "root"

root:unused:11103:10513:U-WINGPFS\root,S-1-5-21-3330551852-1995197583-3793546845-1103:/cygdrive/c/home/root:/bin/bash
```

   If the user is not listed, rebuild your `passwd`:

```
mkpasswd -l -d wingpfs > /etc/passwd
```

4. From the Cygwin shell, run **/usr/bin/ssh-host-config** and respond **yes** to the prompts. When prompted to enter the value of CYGWIN for the daemon, enter **ntsec**. Specify **root** in response to the query for the new user name. You may receive the following warning:

   ```
   ***Warning:  The specified account 'root' does not have the
   ***Warning:  required permissions or group memberships.  This may
   ***Warning:  cause problems if not corrected; continuing...
   ```

   As long as the account (in this case, **root**) is in the local Administrators group, you can ignore this warning.

5. When the installation is complete, enter the following:

   ```
   $ net start sshd

   The CYGWIN sshd service is starting.
   The CYGWIN sshd service was started successfully.
   ```

Note: The OpenSSH READMEs are available at `/usr/share/doc/openssh`. Also see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

Once OpenSSH is installed, the GPFS administrative account **root** needs to be configured so that it can issue **ssh** and **scp** commands without requiring a password and without producing any extraneous messages. This kind of passwordless access is required from any node used for GPFS administration to all other nodes in the cluster.

For additional information, see *Requirements for administering a GPFS file system* in *IBM Spectrum Scale: Administration and Programming Reference* and *Troubleshooting Windows problems* in *IBM Spectrum Scale: Problem Determination Guide*.

Configuring the Windows HPC server

In order for Windows HPC Server to function with GPFS, disable dynamic hosts file and re-enable dynamic DNS updates by doing the following

1. On all nodes in the Windows HPC cluster, open the hosts file **%systemroot%\system32\drivers\etc\hosts** and change **ManageFile = true** to **ManageFile = false**.
2. On the HPC head node, execute the following from HPC PowerShell, assuming all nodes are part of the head node's Active Directory domain.

   Go to **Start > All Programs > Microsoft HPC Pack > HPC PowerShell** and execute:

   ```
   Set-HpcNetwork -EnterpriseDnsRegistrationType WithConnectionDnsSuffix
   ```

Chapter 6. Migration, coexistence and compatibility

To migrate to GPFS 4.1, first consider whether you are migrating from GPFS 3.5 or from an earlier release of GPFS, and then consider coexistence and compatibility issues.

GPFS supports a limited form of backward compatibility between two adjacent GPFS releases. Limited backward compatibility allows you to temporarily operate with a mixture of GPFS 4.1 and GPFS 3.5 nodes:

- Within a cluster this enables you to perform a rolling upgrade to the new GPFS 4.1 version of the code provided your current version is GPFS 3.5.
- In a multicluster environment this allows the individual clusters to be upgraded on their own schedules. Access to the file system data can be preserved even though some of the clusters may still be running GPFS 3.5 level.

Rolling upgrades allow you to install new GPFS code one node at a time without shutting down GPFS on other nodes. However, you must upgrade all nodes within a short time. The time dependency exists because some GPFS 4.1 features become available on each node as soon as the node is upgraded, while other features will not become available until you upgrade all participating nodes. Once all nodes have been migrated to the new code, you must finalize the migration by running the commands **mmchconfig release=LATEST** and **mmchfs -V full** (or **mmchfs -V compat**). Also, certain new features may require you to run the **mmmigratefs** command to enable them.

Attention: Starting with GPFS 4.1, a full backup (**-t full**) with **mmbackup** is required if a full backup has never been performed with GPFS 3.3 or later. For more information, see *File systems backed up using GPFS 3.2 or earlier versions of mmbackup* in *IBM Spectrum Scale: Administration and Programming Reference*.

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

GPFS migration consists of these topics:

Migrating to IBM Spectrum Scale 4.2.x from IBM Spectrum Scale 4.1.x

IBM Spectrum Scale 4.2 release supports node-at-a-time migration if the previous nodes in the cluster are running IBM Spectrum Scale 4.1.x release. IBM Spectrum Scale 4.1 nodes can coexist and interoperate with nodes that are running IBM Spectrum Scale 4.2. However, new functions that depend on format changes will not be available until all nodes have been migrated.

© Copyright IBM Corp. 2014, 2016

Note: To migrate to IBM Spectrum Scale 4.2.0.1 release, the user must have the IBM Spectrum Scale 4.2 release installed and running on the system. Before proceeding towards the upgrade, see the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).

To migrate a cluster to IBM Spectrum Scale 4.2.x release from IBM Spectrum Scale 4.1 release, perform the following steps:

1. Stop all user activity in the file systems on the designated node.

   For information about how to stop and unmount NFS running over IBM Spectrum Scale file systems, see *Unmounting a file system after NFS export* in *IBM Spectrum Scale: Administration and Programming Reference*.
2. Cleanly unmount the mounted IBM Spectrum Scale file system. Do not use force to unmount the designated node.
3. Follow any local administrative backup procedures to ensure protection of your file system data in the event of a failure.
4. Stop IBM Spectrum Scale on the node to be migrated in the cluster using the **mmshutdown** command.

   For example: **mmshutdown -N k164n04**, where k164n04 is the designated node.
5. Migrate to IBM Spectrum Scale 4.2 release depending on the operating system:

   For Linux nodes:

   - Extract the IBM Spectrum Scale software as described in the "Extracting the GPFS software on Linux nodes" on page 103 topic.

     The installable images will be extracted to the following location: /usr/lpp/mmfs/4.2.0.0

     For SLES and RedHat Enterprise Linux nodes:

     – For IBM All Spectrum Scale Editions: **rpm -Fvh /usr/lpp/mmfs/4.2.0.0/gpfs*.rpm**

     – For IBM Spectrum Scale Advanced Edition:

       rpm -ivh gpfs.adv-4.2.0-0*.rpm - This must be installed to use the Async disaster recovery feature.

       rpm -ivh gpfs.crypto-4.2.0-0*.rpm

     – For IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition:

       To optionally install the call home feature, see *Understanding the call home functionality* in *IBM Spectrum Scale: Advanced Administration Guide*.

       To optionally install the Performance Monitoring tool, see the "Manually installing the Performance Monitoring tool" on page 139 topic.

       To optionally install the IBM Spectrum Scale GUI, see the "Manually installing IBM Spectrum Scale management GUI" on page 144 topic.

       Note: When migrating to the IBM Spectrum Scale 4.2 from IBM Spectrum Scale 4.1.1, RHEL7 users can refer to the "Upgrading GPFS components with the spectrumscale installation toolkit" on page 128 topic for more information on deploying protocols using the **spectrumscale** installation toolkit.

   - For Debian and Ubuntu Linux users:

     – For IBM Spectrum Scale Express Edition, run the following command:

       ```
       dpkg -i gpfs.base*deb gpfs.gpl*deb gpfs.gskit*deb
       gpfs.msg*deb gpfs.doc*deb
       ```

     – For IBM Spectrum Scale Standard Edition, run the following command:

       ```
       dpkg -i gpfs.base*deb gpfs.gpl*deb gpfs.gskit*deb
       gpfs.msg*deb gpfs.doc*deb gpfs.ext*deb
       ```

     – For IBM Spectrum Scale Advanced Edition, run the following command:

       ```
       dpkg -i gpfs.base*deb gpfs.gpl*deb gpfs.gskit*deb gpfs.msg*deb
       gpfs.doc*deb gpfs.ext*deb  gpfs.adv*deb gpfs.crypto*deb
       ```

– To optionally install the Performance Monitoring tool on Debian 7, run the following command:

```
dpkg -i gpfs.gss.pmcollector-4.2.*.D7*deb
gpfs.gss.pmsensors-4.2.*D7*.deb
```

– To optionally install the Performance Monitoring tool on Ubuntu 14, run the following command:

```
dpkg -i gpfs.gss.pmsensors_4.2.*.U14.*.deb
gpfs.gss.pmcollector_4.2.*.U14.*.deb
```

- After successful migration, you should rebuild the GPL layer. For more information, see the "Building the GPFS portability layer on Linux nodes" on page 149 topic.

For Hadoop users: To upgrade the IBM Spectrum Scale Hadoop Connector, see *Upgrading IBM Spectrum Scale connector* in *IBM Spectrum Scale: Advanced Administration Guide*.

For AIX nodes:

- Copy the installation images and install the IBM Spectrum Scale licensed programs as described in the Chapter 4, "Installing IBM Spectrum Scale on AIX nodes," on page 153 section.

  Note: If you use SMIT to migrate IBM Spectrum Scale on AIX, IBM Spectrum Scale Advanced Edition users must install `gpfs.adv`.

For Windows nodes

a. Open the **Programs and Features** control panel and remove **IBM Spectrum Scale Express Edition 4.1.1.X** or **IBM Spectrum Scale Standard Edition 4.1.1.X**.
   1) Uninstall IBM Spectrum Scale 4.1.1 and reboot.
   2) Uninstall IBM GPFS GSKit 8.0.x.x and reboot.
   3) Uninstall the IBM Spectrum Scale 4.1.1 license.

b. Copy the installation images and install the IBM Spectrum Scale licensed program as described in Installing IBM Spectrum Scale on Windows nodes.

   Note: For Windows migrations, it is required that all IBM Spectrum Scale administration commands are executed from the 4.2 node.

6. Start IBM Spectrum Scale on the designated node in the cluster using the **mmstartup** command. For example: **mmstartup -N k164n04**, where `k164n04` is the designated node.
7. Mount the file systems if this is not done automatically when the IBM Spectrum Scale daemon starts.

When all nodes in the cluster have been successfully migrated to the new IBM Spectrum Scale level, proceed to the Completing the migration to a new level of IBM Spectrum Scale topic.

Migrating to IBM Spectrum Scale 4.1.1.x from GPFS V4.1.0.x

Use the following information to migrate from V4.1.0.x to V4.1.1.x.

For Linux:

1. Prior to upgrading GPFS on a node, all applications that depend on GPFS (For example, DB2®) must be stopped. Any GPFS file systems that are NFS exported must be unexported prior to unmounting GPFS file systems.
2. Stop GPFS on the node. Verify that the GPFS daemon has terminated and that the kernel extensions have been unloaded (**mmfsenv -u**). If the command **mmfsenv -u** reports that it cannot unload the kernel extensions because they are "busy", then the install can proceed, but the node must be rebooted after the install. By "busy" this means that some process has a "current directory" in some GPFS file system directory or has an open file descriptor. The freeware program lsof can identify the process and the process can then be killed. Retry **mmfsenv -u** and if that succeeds then a reboot of the node can be avoided.
3. Upgrade GPFS using the RPM command as follows (make sure you are in the same directory as the files) :

For SLES or RHEL systems

```
rpm -Fvh gpfs*.rpm
```

For Debian systems

```
dpkg -i gpfs*.deb
```

4. You can verify the installation of the GPFS SLES or RHEL Linux RPMs on each node. To check that the software has been successfully installed, use the **rpm** command:

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

   The system should return output similar to the following:

   ```
   gpfs.gpl-4.1.1-1
   gpfs.docs-4.1.1-1
   gpfs.msg.en_US-4.1.1-1
   gpfs.base-4.1.1-1
   gpfs.gskit-8.0.50-40
   ```

   If you have the GPFS Standard Edition or the GPFS Advanced Edition installed, you should also see the following line in the output:

   ```
   gpfs.ext-4.1.1-1
   ```

   If you have the GPFS Advanced Edition installed, you should also see the following line in the output:

   ```
   gpfs.crypto-4.1.1-1
   ```

5. You can verify the installation of the GPFS Debian Linux packages on each node. To check that the software has been successfully installed, use the **dpkg** command:

   ```
   dpkg -l | grep gpfs
   ```

   The system should return output similar to the following:

   ```
   ii gpfs.base 4.1.1-1     GPFS File Manager
   ii gpfs.docs 4.1.1-1      GPFS Server Manpages and Documentation
   ii gpfs.gpl 4.1.1-1         GPFS Open Source Modules
   ii gpfs.gskit 8.0.50-40   GPFS GSKit Cryptography Runtime
   ii gpfs.msg.en_US 4.1.1-1 GPFS Server Messages - U.S. English
   ```

   If you have the GPFS Standard Edition or the GPFS Advanced Edition installed, you should also see the following line in the output:

   ```
   ii gpfs.ext 4.1.1-1      GPFS Extended Features
   ```

   If you have the GPFS Advanced Edition installed, you should also see the following line in the output:

   ```
   ii gpfs.crypto 4.1.1-1    GPFS Cryptographic Subsystem
   ```

6. Recompile any GPFS portability layer modules you may have previously compiled using the **mmbuildgpl** command.

For AIX:

1. Prior to upgrading GPFS on a node, all applications that depend on GPFS (For example, DB2) must be stopped. Any GPFS file systems that are NFS exported must be unexported prior to unmounting GPFS file systems.
2. To migrate directly from 4.1.0.x to 4.1.1.x, you must put all Unnnnnn.gpfs*.bff images from 4.1.1.0 and 4.1.1.x in the same directory prior to running the **installp** command or SMIT.
3. Stop GPFS on the node. Verify that the GPFS daemon has terminated and that the kernel extensions have been unloaded (**mmfsenv -u**). If the command **mmfsenv -u** reports that it cannot unload the kernel extensions because they are "busy", then the install can proceed, but the node must be rebooted after the install. By "busy" this means that some process has a "current directory" in some GPFS

filesystem directory or has an open file descriptor. The freeware program **lsof** can identify the process and the process can then be killed. Retry **mmfsenv -u** and if that succeeds then a reboot of the node can be avoided.

4. Upgrade GPFS using the **installp** command or via SMIT on the node. If you are in the same directory as the install packages, an example command might be:

   ```
   installp -agXYd . gpfs
   ```

5. You can verify that the installation procedure placed the required GPFS files on each node by running the **lslpp** command

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

   The system should return output similar to the following:

   ```
   Fileset                 Level      State        Description
   ----------------------------------------------------------------------------
   Path: /usr/lib/objrepos
   gpfs.base              4.1.1-1 COMMITTED GPFS File Manager
   gpfs.crypto            4.1.1-1 COMMITTED GPFS Cryptographic Subsystem
   gpfs.ext               4.1.1-1 COMMITTED GPFS Extended Features
   gpfs.gskit            8.0.50.40 COMMITTED GPFS GSKit Cryptography Runtime
   gpfs.msg.en_US         4.1.1-1 COMMITTED GPFS Server Messages - U.S. English
   Path: /etc/objrepos
   gpfs.base              4.1.1-1 COMMITTED GPFS File Manager
   Path: /usr/share/lib/objrepos
   gpfs.docs.data         4.1.1-1 COMMITTED GPFS Server Manpages and
   Documentation
   ```

   The output that is returned on your system can vary depending on if you have the GPFS Express Edition, GPFS Standard Edition, or GPFS Advanced Edition installed.

For Windows:

1. Downloaded the GPFS 4.1.1-x update package into any directory on your system.

   Note: This update requires a prior level of GPFS version 4.1.x on your system. In such a case, you need to simply uninstall the previous 4.1.x level and proceed with installing this upgrade. Upgrading from a prior 4.1 level does not require installing the GPFS license package again. However if you are upgrading directly from version 3.5 or prior (any level), or installing version 4.1 for the first time on your system, you must first install the GPFS license package (gpfs.base-4.1-Windows-license.msi) before installing this update.

2. Extract the contents of the ZIP archive so that the .msi file it includes is directly accessible to your system.
3. Uninstall the system's current version of GPFS using the Programs and Features control panel. If prompted to reboot the system, do this before installing the update package.

Migrating to IBM Spectrum Scale V4.2 from GPFS V3.5

Node-at-a-time migration is not available when you migrate a cluster to IBM Spectrum Scale V4.2 from GPFS V3.5. You must shut down the cluster and migrate all the nodes at once. If this method is not acceptable, you might consider migrating in two steps: first migrate the cluster from GPFS V3.5 to GPFS V4.1, and then migrate from GPFS V4.1 to IBM Spectrum Scale 4.2.

To migrate a cluster to Spectrum Scale V4.2 from GPFS V3.5, follow these steps:

1. Stop all user activity in the file systems.

   For information about how to stop and then unmount NFS running over GPFS file systems, see *Unmounting a file system after NFS export* in *IBM Spectrum Scale: Administration and Programming Reference*.

2. Cleanly unmount the mounted GPFS file system. Do not use force unmount.

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

3. Follow any local administrative backup procedures to ensure protection of your file system data in the event of a failure.
4. Stop GPFS on all nodes in the cluster. For example:

   ```
   mmshutdown -a
   ```
5. For each node, run the appropriate uninstall program
   - For example, for Linux nodes:

     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.msg.en_US gpfs.ext gpfs.crypto
     ```
   - For AIX nodes:

     ```
     installp -u gpfs
     ```
   - For Windows nodes, open the "Uninstall or change a program window" by clicking **Start** > **Control Panel** > **Programs and Features** and uninstall the IBM General Parallel File System.

     Note: To upgrade a GPFS V3.5 Windows node (SUA-based) to IBM Spectrum Scale Standard Edition V4.2 (Cygwin 64-bit based), follow these steps:

     a. Uninstall GPFS V3.5 and reboot.
     b. Uninstall the GPFS V3.5 license.
     c. Disable any SUA daemon, such as OpenSSH, that might have been configured. Do not uninstall SUA yet, or you might lose GPFS configuration information.
     d. Install the 64-bit version of Cygwin. For more information, see the help topic “Installing the 64-bit version of Cygwin” on page 164.
     e. Copy the installation images and install the IBM Spectrum Scale licensed program as described in Chapter 5, “Installing IBM Spectrum Scale on Windows nodes,” on page 157.
     f. Uninstall SUA completely.
6. For each node, copy the installation images and install the IBM Spectrum Scale licensed program as described in “Installing GPFS on Linux nodes” on page 102, Chapter 4, “Installing IBM Spectrum Scale on AIX nodes,” on page 153, or Chapter 5, “Installing IBM Spectrum Scale on Windows nodes,” on page 157.
7. Start GPFS on the cluster. For example:

   ```
   mmstartup -a
   ```
8. Mount the file systems if they are not mounted automatically when the GPFS daemon starts.

When you have successfully migrated all nodes in the cluster to the new level, see the help topic “Completing the migration to a new level of IBM Spectrum Scale” on page 178.

Migrating to IBM Spectrum Scale 4.2 from GPFS 3.4 or GPFS 3.3

Node-at-a-time migration is not available when migrating to IBM Spectrum Scale 4.2 from GPFS 3.4 or GPFS 3.3. The cluster must be completely shut down and *all* nodes migrated at the same time. If this is not acceptable, and your current level is GPFS 3.4, you may want to consider an intermediate migration to GPFS 3.5 first.

To migrate a cluster to IBM Spectrum Scale 4.2 from GPFS 3.4 or GPFS 3.3, perform these steps:

1. Stop all user activity in the file systems.

   For information about how to stop and then unmount NFS running over GPFS file systems, see *Unmounting a file system after NFS export* in *IBM Spectrum Scale: Administration and Programming Reference*.
2. Cleanly unmount the mounted GPFS file system. Do not use force unmount.
3. Follow any local administrative backup procedures to ensure protection of your file system data in the event of a failure.
4. Stop GPFS on all nodes in the cluster:

```
mmshutdown -a
```

5. Run the appropriate de-installation program to remove GPFS from each node in the cluster. For example:
   - For Linux nodes:
     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.msg.en_US
     ```
   - For AIX nodes:
     ```
     installp -u gpfs
     ```
   - For Windows nodes, open the **Programs and Features** control panel and remove **IBM General Parallel File System**.

     Note: To upgrade a GPFS 3.4 Windows node (SUA based) to GPFS 4.1 Standard Edition (Cygwin 64-bit based), proceed as follows:
     a. Uninstall GPFS 3.4 and reboot.
     b. Uninstall the GPFS 3.4 license.
     c. Disable any SUA daemon, such as OpenSSH, that might have been configured. Do *not* uninstall SUA yet, or you may lose GPFS configuration information.
     d. Install the 64-bit version of Cygwin. See "Installing the 64-bit version of Cygwin" on page 164 for more information.
     e. Install `gpfs.ext-4.1.1-Windows-license.msi`.
     f. Install `gpfs.ext-4.1.1.x-Windows.msi`
     g. Install IBM GSKit for GPFS.
     h. Uninstall SUA completely.
6. Copy the installation images and install the GPFS product on each node in the cluster as described in "Installing GPFS on Linux nodes" on page 102, Chapter 4, "Installing IBM Spectrum Scale on AIX nodes," on page 153, or Chapter 5, "Installing IBM Spectrum Scale on Windows nodes," on page 157.

   Note: For Windows migrations, it is required that all IBM Spectrum Scale 4.2 administration commands are executed from the 4.2 node.
7. Start GPFS on all nodes in the cluster:
   ```
   mmstartup -a
   ```
8. Mount the file systems if this is not done automatically when the GPFS daemon starts.
9. Proceed to "Completing the migration to a new level of IBM Spectrum Scale" on page 178.

Migrating to IBM Spectrum Scale 4.2 from GPFS 3.2 or earlier releases of GPFS

Starting with GPFS 4.1, clusters running GPFS 3.2 and earlier releases are no longer supported, so your cluster must be recreated. If this is not acceptable, you may want to consider an intermediate migration to GPFS 3.5, or any other supported release, first.

To migrate your GPFS cluster to IBM Spectrum Scale 4.2 from GPFS 3.2 or earlier, follow these steps:

1. Stop all user activity in the file systems.
2. Follow any local administrative backup procedures to ensure protection of your file system data in the event of a failure.
3. Cleanly unmount all mounted GPFS file systems. Do not use force unmount.
4. Shut down the GPFS daemon on all nodes in the cluster:
   ```
   mmshutdown -a
   ```
5. Export the GPFS file systems by issuing the **mmexportfs** command:
   ```
   mmexportfs all -o exportDataFile
   ```

This command creates the configuration output file *exportDataFile*, which contains all exported configuration data. Retain this file because it is required when issuing the **mmimportfs** command to import your file systems into the new cluster or in the event that you decide to go back to the previous release.

6. Record any site-specific configuration attributes that are currently in effect and that you want to preserve.
7. Delete all existing nodes by issuing this command:

   ```
   mmdelnode -a
   ```
8. Run the appropriate de-installation program to remove GPFS from each node in the cluster. For example:
   - For Linux nodes:

     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.msg.en_US
     ```
   - For AIX nodes:

     ```
     installp -u gpfs
     ```
   - For Windows nodes, open the **Programs and Features** control panel and remove **IBM General Parallel File System**.
9. Copy the installation images and install the GPFS product on each node in the cluster as described in "Installing GPFS on Linux nodes" on page 102, Chapter 4, "Installing IBM Spectrum Scale on AIX nodes," on page 153, or Chapter 5, "Installing IBM Spectrum Scale on Windows nodes," on page 157.
10. Decide which nodes in your system will be quorum nodes (see "Quorum" on page 41).
11. Create a new GPFS cluster across all desired nodes by issuing the **mmcrcluster** command. Run the **mmchlicense** command to set the appropriate licenses after the cluster is created.
12. Using the **mmchconfig** command, restore your site-specific configuration attributes from step 6.
13. To complete the movement of your file systems to the new cluster, using the configuration file created in step 5 on page 177, issue:

    ```
    mmimportfs all -i exportDataFile
    ```
14. Start GPFS on all nodes in the new cluster:

    ```
    mmstartup -a
    ```
15. Mount the file systems if this is not done automatically when the GPFS daemon starts.
16. Proceed to "Completing the migration to a new level of IBM Spectrum Scale."

Completing the migration to a new level of IBM Spectrum Scale

You should operate the cluster with the new level of the IBM Spectrum Scale until you are sure that you want to permanently migrate to the new level.

When you are ready to migrate the cluster permanently to the new level, you must complete the migration of both the cluster configuration data and all the file systems. If you decide not to migrate to the new level, you can revert to the previous level. For more information, see "Reverting to the previous level of GPFS" on page 180.

After you have migrated all nodes to the latest licensed version of IBM Spectrum Scale, follow these steps to complete the migration:

1. Verify that the **cipherList** configuration variable and the SHA message digest are set to valid values. The **cipherList** variable specifies the security mode for communications among nodes in the cluster and with nodes in other clusters. The SHA message digest is a hash result generated by a cryptographic hash function. Follow these steps:
   a. On the command line, enter the command `mmauth show .` to display the current values. The following listing shows the command and example output:

```
# mmauth show .
Cluster name:        zounds.cluster (this cluster)
Cipher list:         (none specified)
SHA digest:          (undefined)
File system access:  (all rw)
```

b. If the value for the `SHA digest` is `(undefined)`, enter the following command to generate a public/private key pair and a SHA message digest:

```
mmauth genkey new
```

Enter `mmauth show .` again as before and verify that the value for `SHA digest` is no longer `(undefined)`.

c. If the value for `Cipher list` is `(none specified)`, enter the following command to set it to **AUTHONLY**:

```
mmauth update . -l AUTHONLY
```

Enter `mmauth show .` again as before and verify that the value for `Cipher list` is no longer `(none specified)`.

2.

Enter the following command to migrate the cluster configuration data and enable new functionality throughout the cluster:

```
mmchconfig release=LATEST
```

The **mmchconfig** command lists the names of any nodes that are not available or cannot be reached. If any such nodes are listed, correct the problem and run the command again until it verifies all the nodes and completes successfully.

Important: The command fails if `Cipher list` is set to a security level lower than **AUTHONLY**. The only lower security level is **EMPTY**. If you want to stay at this level, run the following command:

```
mmchconfig release=LATEST --accept-empty-cipherlist-security
```

3.

If you have not already done so, assign an appropriate GPFS license to each of the nodes in the cluster. See "IBM Spectrum Scale license designation" on page 80 for a detailed description of the GPFS license types. To see what the minimum required GPFS license is for each of the nodes in the cluster, enter the following command:

```
mmlslicense -L
```

To assign a GPFS server license to the nodes that require it, enter the following command:

```
mmchlicense server -N NodeList
```

To assign a GPFS client license to the nodes that require it, issue:

```
mmchlicense client -N NodeList
```

4. Enable backward-compatible format changes or migrate all file systems to the latest metadata format changes.

Attention: Before continuing with this step, it is important to understand the differences between **mmchfs -V compat** and **mmchfs -V full**:

- If you issue **mmchfs -V compat**, only changes that are backward compatible with GPFS 3.5 will be enabled. Nodes in remote clusters that are running GPFS 3.5 will still be able to mount the file system. Nodes running GPFS 3.4 or earlier will no longer be able to mount the file system.
- If you issue **mmchfs -V full**, all new functions that require different on-disk data structures will be enabled. Nodes in remote clusters running an older GPFS version will no longer be able to mount the file system. If there are any nodes running an older GPFS version that have the file system mounted at the time this command is issued, the **mmchfs** command will fail.

To enable backward-compatible format changes, issue the following command:

```
mmchfs FileSystem -V compat
```

To migrate all file systems to the latest metadata format changes, issue the following command:

```
mmchfs FileSystem -V full
```

Certain new file system features may require additional processing that cannot be handled by the **mmchfs -V** command alone. To fully activate such features, in addition to **mmchfs -V**, you will also have to run the **mmmigratefs** command. An example of such a feature is enabling fast extended attributes for file systems older than GPFS 3.4.

Note: The first mount of a file system after running **mmchfs -V** might fail with a no-disk-space error. This might occur if the file system is relatively low on metadata disk space (10% or less free). The user should be able to mount the file system without problem after the initial failed mount.

To activate fast extended attributes, issue the following command:

```
mmmigratefs FileSystem --fastea
```

5. If you use the **mmbackup** command to back up your file system, a full backup may be required if a full backup has never been performed with GPFS 3.3 or later. For more information, see *File systems backed up using GPFS 3.2 or earlier versions of mmbackup* in *IBM Spectrum Scale: Administration and Programming Reference*.

Reverting to the previous level of GPFS

If you should decide not to continue the migration to the latest level of GPFS, and you have not yet issued the **mmchfs -V** command, you can reinstall the earlier level of GPFS.

Important: Once a file system has been migrated explicitly by issuing the **mmchfs -V full** command, the disk images can no longer be read by a prior version of GPFS. You will be required to re-create the file system from the backup media and restore the content if you choose to go back after this command has been issued. The same rules apply for file systems that are newly created with GPFS 4.2.

You can revert back to GPFS 4.1.x.

If you have performed backups with the **mmbackup** command using the 4.2 version and decide to revert to an earlier version, you must rebuild the **mmbackup** shadow database using the **mmbackup** command with either the **-q** or **--rebuild** option.

The procedure differs depending on whether you have issued the **mmchconfig release=LATEST** command or not.

Reverting to a previous level of GPFS when you have *not* issued mmchconfig release=LATEST

If you have **not** issued the **mmchconfig release=LATEST** command, perform these steps.

1. Stop all user activity in the file systems.
2. Cleanly unmount all mounted GPFS file systems. Do not use force unmount.
3. Stop GPFS on all nodes in the cluster:

   ```
   mmshutdown -a
   ```

4. Run the appropriate uninstallation program to remove GPFS from each node in the cluster. For example:
   - For Linux nodes (this example is only applicable on the IBM Spectrum Scale Express Edition):

     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.gskit gpfs.msg.en_US
     ```

   - For AIX nodes:

     ```
     installp -u gpfs
     ```

   - For Windows nodes, open the **Programs and Features** control panel and remove **IBM General Parallel File System**.

For the remaining steps, see IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html), and search for the appropriate *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* for your release.

5. Copy the installation images of the previous GPFS licensed program on all affected nodes.
6. Install the original install images and all required PTFs.
7. For Linux nodes running GPFS, you must rebuild the GPFS portability layer.
8. Reboot all nodes.

Reverting to a previous level of GPFS when you *have* issued mmchconfig release=LATEST

If you *have* issued the **mmchconfig release=LATEST** command, you must rebuild the cluster. Perform these steps.

1. Stop all user activity in the file systems.
2. Cleanly unmount all mounted GPFS file systems. Do not use force unmount.
3. Stop GPFS on all nodes in the cluster:
   ```
   mmshutdown -a
   ```
4. Export the GPFS file systems by issuing the **mmexportfs** command:
   ```
   mmexportfs all -o exportDataFile
   ```
5. Delete the cluster:
   ```
   mmdelnode -a
   ```
6. Run the appropriate de-installation program to remove GPFS from each node in the cluster. For example:
   - For Linux nodes:
     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs gpfs.msg.en_US
     ```
   - For AIX nodes:
     ```
     installp -u gpfs
     ```
   - For Windows nodes, open the **Programs and Features** control panel and remove **IBM General Parallel File System**.

   For the remaining steps, see IBM Spectrum Scale in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/ibmspectrumscale_welcome.html), and search for the appropriate *IBM Spectrum Scale: Concepts, Planning, and Installation Guide* for your release.
7. Copy the installation images of the previous GPFS licensed program on all affected nodes.
8. Install the original installation images and all required PTFs.
9. For Linux nodes running GPFS, you must rebuild the GPFS portability layer.
10. Reboot all nodes.
11. Recreate your original cluster using the **mmcrcluster** command. Run the **mmchlicense** command to set the appropriate licenses after the cluster is created.
12. Use the **mmchconfig** command to restore any previously set configuration settings that are compatible with GPFS 4.1 or below.
13. Import the file system information using the **mmimportfs** command. Specify the file created by the **mmexportfs** command from Step 4:
    ```
    mmimportfs all -i exportDataFile
    ```
14. Start GPFS on all nodes in the cluster, issue:
    ```
    mmstartup -a
    ```
15. Mount the file systems if this is not done automatically when the GPFS daemon starts.

Coexistence considerations

Each GPFS cluster can have multiple GPFS file systems that coexist on the cluster, but function independently of each other. In addition, each file system might have different data management programs.

Note: The GPFS Data Management API (DMAPI) and GPFS file system snapshots can coexist; however, access to the files in a snapshot using DMAPI is restricted. For more information, see *IBM Spectrum Scale: Data Management API Guide*.

Compatibility considerations

All applications that ran on the previous release of GPFS will run on the new level of GPFS. File systems that were created under the previous release of GPFS can be used under the new level of GPFS.

Considerations for Tivoli Storage Manager for Space Management

Migrating to GPFS 3.3 or beyond requires consideration for Tivoli Storage Manager for Space Management. Tivoli Storage Manager for Space Management requires that all nodes in a cluster are configured with a DMAPI file handle size of 32 bytes.

During migration, it is possible to have older versions of GPFS that have a DMAPI file handle size of 16 bytes. Until all nodes in the cluster have been updated to the latest release and the DMAPI file handle size has been changed to 32 bytes, IBM Tivoli Storage Manager for Space Management must be disabled. Run **mmlsconfig dmapiFileHandleSize** to see what value is set.

After all nodes in the cluster are upgraded to the latest release and you change the DMAPI file handle size to 32 bytes, you can enable IBM Tivoli Storage Manager for Space Management.

The DMAPI file handle size is configured with the **dmapiFileHandleSize** option. For more information about this option, see *GPFS configuration attributes for DMAPI* in *IBM Spectrum Scale: Data Management API Guide*.

Applying maintenance to your GPFS system

Before applying maintenance to your GPFS system, there are several things you should consider.

Remember that:

1. There is limited interoperability between GPFS 4.2 nodes and nodes running GPFS 4.1 This function is intended for short-term use. You will not get the full functions of GPFS 4.2 until all nodes are using GPFS 4.2.
2. Interoperability between maintenance levels will be specified on a per-maintenance-level basis. See the IBM Spectrum Scale FAQ in IBM Knowledge Center (www.ibm.com/support/knowledgecenter/STXKQY/gpfsclustersfaq.html).
3. Maintenance images for GPFS Linux retrieved from the web are named differently from the installation images of the GPFS Linux product. Maintenance images contain the word **update** in their name, for example: **gpfs.base-4.2.2-0.x86_64.update.rpm**.
4. When applying maintenance, the GPFS file systems will need to be unmounted and GPFS shut down on the node being upgraded. Any applications using a GPFS file system should be shut down before applying maintenance to avoid application errors or possible system halts.

   Note: If the AIX Alternate Disk Install process is used, the new image is placed on a different disk. The running install image is not changed, and GPFS is not shut down. The new image is not used until the node is rebooted.

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

To download fixes, go to the IBM Support Portal: Downloads for General Parallel File System (www.ibm.com/support/entry/portal/Downloads/Software/Cluster_software/General_Parallel_File_System) and obtain the fixes for your hardware and operating system.

To install the latest fixes:

- For Linux nodes, see "Installing GPFS on Linux nodes" on page 102.
- For AIX nodes, see Chapter 4, "Installing IBM Spectrum Scale on AIX nodes," on page 153.
- For Windows nodes, see Chapter 5, "Installing IBM Spectrum Scale on Windows nodes," on page 157.

Chapter 7. Configuring and tuning your system for GPFS

In addition to configuring your GPFS cluster, you need to configure and tune your system.

For more information, see *GPFS cluster creation considerations* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

Values suggested here reflect evaluations made at the time this documentation was written. For the latest system configuration and tuning settings, 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)).

Additional GPFS and system configuration and tuning considerations include:

1. "General system configuration and tuning considerations"
2. "Linux configuration and tuning considerations" on page 189
3. "AIX configuration and tuning considerations" on page 191
4. "Configuring Windows" on page 162

For more information, see:

- *Monitoring GPFS I/O performance with the* **mmpmon** *command* in *IBM Spectrum Scale: Advanced Administration Guide*.
- *Using multiple token servers* in *IBM Spectrum Scale: Advanced Administration Guide*.

General system configuration and tuning considerations

You need to take into account some general system configuration and tuning considerations. This topic points you to the detailed information.

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

Configuration and tuning considerations for all systems include:

1. "Clock synchronization"
2. "GPFS administration security"
3. "Cache usage" on page 186
4. "GPFS I/O" on page 188
5. "Access patterns" on page 188
6. "Aggregate network interfaces" on page 188
7. "Swap space" on page 188

Clock synchronization

The clocks of all nodes in the GPFS cluster must be synchronized. If this is not done, NFS access to the data, as well as other GPFS file system operations may be disrupted.

GPFS administration security

Before administering your GPFS file system, make certain that your system has been properly configured for security.

© Copyright IBM Corp. 2014, 2016

This includes:

- Assigning root authority to perform all GPFS administration tasks except:
  - Tasks with functions limited to listing GPFS operating characteristics.
  - Tasks related to modifying individual user file attributes.
- Establishing the authentication method between nodes in the GPFS cluster.
  - Until you set the authentication method, you cannot issue any GPFS commands.
- Designating a remote communication program for remote shell and remote file copy commands.
  - The default remote communication commands are **scp** and **ssh**. You can designate any other remote commands if they have the same syntax.
  - Regardless of which remote commands have been selected, the nodes that you plan to use for administering GPFS must be able to execute commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Cache usage

GPFS creates a number of cache segments on each node in the cluster. The amount of cache is controlled by three attributes.

These attributes have default values at cluster creation time and may be changed through the **mmchconfig** command:

pagepool

The GPFS pagepool is used to cache user data and file system metadata. The pagepool mechanism allows GPFS to implement read as well as write requests asynchronously. Increasing the size of pagepool increases the amount of data or metadata that GPFS can cache without requiring synchronous I/O. The amount of memory available for GPFS pagepool on a particular node may be restricted by the operating system and other software running on the node.

The optimal size of the pagepool depends on the needs of the application and effective caching of its re-accessed data. For systems where applications access large files, reuse data, benefit from GPFS prefetching of data, or have a random I/O pattern, increasing the value for **pagepool** may prove beneficial. However, if the value is set too large, GPFS will start with the maximum that the system allows. See the GPFS log for the value it is running at.

To change the pagepool to 4 GB:

```
mmchconfig pagepool=4G
```

maxFilesToCache

The total number of different files that can be cached at one time. Every entry in the file cache requires some pageable memory to hold the content of the file's inode plus control data structures. This is in addition to any of the file's data and indirect blocks that might be cached in the page pool.

The total amount of memory required for inodes and control data structures can be estimated as:

```
maxFilesToCache × 3 KB
```

Valid values of **maxFilesToCache** range from 1 to 100,000,000. For systems where applications use a large number of files, of any size, increasing the value for **maxFilesToCache** may prove beneficial. This is particularly true for systems where a large number of small files are accessed. The value should be large enough to handle the number of concurrently open files plus allow caching of recently used files.

If the user does not specify a value for **maxFilesToCache**, the default value is 4000.

maxStatCache

This parameter sets aside additional pageable memory to cache attributes of files that are not currently in the regular file cache. This is useful to improve the performance of both the system and GPFS **stat()** calls for applications with a working set that does not fit in the regular file cache.

The memory occupied by the stat cache can be calculated as:

`maxStatCache × 400 bytes`

Valid values of **maxStatCache** range from 0 to 100,000,000. For systems where applications test the existence of files, or the properties of files, without actually opening them (as backup applications do), increasing the value for **maxStatCache** may prove beneficial.

If the user does not specify values for **maxFilesToCache** and **maxStatCache**, the default value of **maxFilesToCache** is 4000, and the default value of **maxStatCache** is 1000.

If the user specifies a value for **maxFilesToCache** but does not specify a value for **maxStatCache**, the default value of **maxStatCache** changes to 4***maxFilesToCache**.

Note: The stat cache is not effective on the Linux platform. Therefore, you need to set the **maxStatCache** attribute to a smaller value, such as 512, on that platform.

The total amount of memory GPFS uses to cache file data and metadata is arrived at by adding **pagepool** to the amount of memory required to hold inodes and control data structures (**maxFilesToCache** × 3 KB), and the memory for the stat cache (**maxStatCache** × 400 bytes) together. The combined amount of memory to hold inodes, control data structures, and the stat cache is limited to 50% of the physical memory on a node running GPFS.

During configuration, you can specify the **maxFilesToCache**, **maxStatCache**, and **pagepool** parameters that control how much cache is dedicated to GPFS. These values can be changed later, so experiment with larger values to find the optimum cache size that improves GPFS performance without negatively affecting other applications.

The **mmchconfig** command can be used to change the values of **maxFilesToCache**, **maxStatCache**, and **pagepool**. The **pagepool** parameter is the only one of these parameters that may be changed while the GPFS daemon is running. A **pagepool** change occurs immediately when using the **-i** option on the **mmchconfig** command. Changes to the other values are effective only after the daemon is restarted.

For further information on these cache settings for GPFS, refer to *GPFS and memory* in *IBM Spectrum Scale: Concepts, Planning, and Installation Guide*.

The GPFS token system's effect on cache settings

Lock tokens play a role in maintaining cache consistency between nodes.

A token allows a node to cache data it has read from disk, because the data cannot be modified elsewhere without revoking the token first. Each token manager can handle approximately 300,000 different file tokens (this number depends on how many distinct byte-range tokens are used when multiple nodes access the same file). If you divide the 300,000 by the number of nodes in the GPFS cluster you get a value that should approximately equal **maxFilesToCache** (the total number of different files that can be cached at one time) + **maxStatCache** (additional pageable memory to cache file attributes that are not currently in the regular file cache).

Note the following about **maxFilesToCache** and **maxStatCache**:

- If the user does not specify values for **maxFilesToCache** and **maxStatCache**, the default value of **maxFilesToCache** is 4000, and the default value of **maxStatCache** is 1000.
- On upgrades to GPFS 4.1 from GPFS 3.4 or earlier, the existing defaults (1000 for **maxFilesToCache** and 4000 for **maxStatCache**) remain in effect.
- If the user specifies a value for **maxFilesToCache** but does not specify a value for **maxStatCache**, the default value of **maxStatCache** changes to 4***maxFilesToCache**.
- **maxFilesToCache** should be large enough to handle the number of concurrently open files plus allow caching of recently used files.

- **maxStatCache** can be set higher on user-interactive-nodes and smaller on dedicated compute-nodes, since **ls -l** performance is mostly a human response issue.
- **maxFilesToCache** and **maxStatCache** are indirectly affected by the **distributedTokenServer** configuration parameter because distributing the tokens across multiple token servers might allow keeping more tokens than if a file system has only one token server.

GPFS I/O

The **maxMBpS** option determines the maximum amount of I/O in MB that can be submitted by GPFS per second. If the default value is not adjusted accordingly it will affect GPFS performance.

Note that setting this number too high can have an adverse effect on performance of the system since overrunning the capabilities of the I/O bus or network adapter tends to drastically degrade throughput. This number is normally set after an empirical study to determine the I/O bandwidth limits of your nodes. The default value is 2048 MB per second.

Access patterns

GPFS attempts to recognize the pattern of accesses (such as strided sequential access) that an application makes to an open file. If GPFS recognizes the access pattern, it will optimize its own behavior.

For example, GPFS can recognize sequential reads and will retrieve file blocks before they are required by the application. However, in some cases GPFS does not recognize the access pattern of the application or cannot optimize its data transfers. In these situations, you may improve GPFS performance if the application explicitly discloses aspects of its access pattern to GPFS through the **gpfs_fcntl()** library call.

Aggregate network interfaces

It is possible to aggregate multiple physical Ethernet interfaces into a single virtual interface. This is known as *Channel Bonding* on Linux and *EtherChannel/IEEE 802.3ad Link Aggregation* on AIX.

GPFS supports using such aggregate interfaces. The main benefit is increased bandwidth. The aggregated interface has the network bandwidth close to the total bandwidth of all its physical adapters. Another benefit is improved fault tolerance. If a physical adapter fails, the packets are automatically sent on the next available adapter without service disruption.

EtherChannel and IEEE802.3ad each requires support within the Ethernet switch. Refer to the product documentation for your switch to determine if EtherChannel is supported.

For details on how to configure EtherChannel and IEEE 802.3ad Link Aggregation and verify whether the adapter and the switch are operating with the correct protocols for IEEE 802.3ad, consult the operating system documentation.

Hint: Make certain that the switch ports are configured for **LACP** (the default is **PAGP**).

For additional service updates regarding the use of EtherChannel:

1. Go to the IBM Support Portal (www.ibm.com/support)
2. In the **Search** box, enter the search term *EtherChannel*
3. Click **Search**

Hint: A useful command for troubleshooting, where device is the Link Aggregation device, is:

```
entstat -d device
```

Swap space

It is highly suggested that a sufficiently large amount of swap space is configured.

While the actual configuration decisions should be made taking into account the memory requirements of other applications, it is suggested to configure at least as much swap space as there is physical memory on a given node.

Linux configuration and tuning considerations

Configuration and tuning considerations for the Linux nodes in your system include the use of the **updatedb** utility, the **vm.min_free_kbytes** kernel tunable, and several other options that can improve GPFS performance.

For the latest system configuration and tuning settings, 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)).

For more configuration and tuning considerations for Linux nodes, see the following topics:

1. "updatedb considerations"
2. "Memory considerations"
3. "GPFS helper threads" on page 190
4. "Communications I/O" on page 190
5. "Disk I/O" on page 190

updatedb considerations

On some Linux distributions, the system is configured by default to run the file system indexing utility **updatedb** through the **cron** daemon on a periodic basis (usually daily).

This utility traverses the file hierarchy and generates a rather extensive amount of I/O load. For this reason, it is configured by default to skip certain file system types and nonessential file systems. However, the default configuration does not prevent **updatedb** from traversing GPFS file systems. In a cluster this results in multiple instances of **updatedb** traversing the same GPFS file system simultaneously. This causes general file system activity and lock contention in proportion to the number of nodes in the cluster. On smaller clusters, this may result in a relatively short-lived spike of activity, while on larger clusters, depending on the overall system throughput capability, the period of heavy load may last longer. Usually the file system manager node will be the busiest, and GPFS would appear sluggish on all nodes. Re-configuring the system to either make **updatedb** skip all GPFS file systems or only index GPFS files on one node in the cluster is necessary to avoid this problem.

Memory considerations

It is recommended that you adjust the **vm.min_free_kbytes** kernel tunable. This tunable controls the amount of free memory that Linux kernel keeps available (that is, not used in any kernel caches).

When **vm.min_free_kbytes** is set to its default value, on some configurations it is possible to encounter memory exhaustion symptoms when free memory should in fact be available. Setting **vm.min_free_kbytes** to a higher value (Linux **sysctl** utility could be used for this purpose), on the order of magnitude of 5-6% of the total amount of physical memory, should help to avoid such a situation.

See the GPFS Redbooks® papers for more information:

- *GPFS Sequential Input/Output Performance on IBM pSeries 690* (www.redbooks.ibm.com/redpapers/pdfs/redp3945.pdf)
- *Native GPFS Benchmarks in an Integrated p690/AIX and x335/Linux Environment* (www.redbooks.ibm.com/redpapers/pdfs/redp3962.pdf)

GPFS helper threads

GPFS uses helper threads, such as **prefetchThreads** and **worker1Threads**, to improve performance.

Since systems vary, it is suggested you simulate an expected workload in GPFS and examine available performance indicators on your system. For instance some SCSI drivers publish statistics in the **/proc/scsi** directory. If your disk driver statistics indicate that there are many *queued requests* it may mean you should throttle back the helper threads in GPFS.

For more information, see the article "Tuning Parameters" in the IBM Spectrum Scale wiki in developerWorks®. See https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/General%20Parallel%20File%20System%20%28GPFS%29/page/Tuning%20Parameters.

Communications I/O

Values suggested here reflect evaluations made at the time this documentation was written. For the latest system configuration and tuning settings, see the IBM Spectrum Scale Wiki (www.ibm.com/developerworks/community/wikis/home/wiki/General Parallel File System (GPFS)).

To optimize the performance of GPFS and your network, it is suggested you do the following:

- Enable Jumbo Frames if your switch supports it.

  If GPFS is configured to operate over Gigabit Ethernet, set the MTU size for the communication adapter to 9000.
- Verify **/proc/sys/net/ipv4/tcp_window_scaling** is enabled. It should be by default.
- Tune the TCP window settings by adding these lines to the **/etc/sysctl.conf** file:

```
# increase Linux TCP buffer limits
net.core.rmem_max = 8388608
net.core.wmem_max = 8388608
# increase default and maximum Linux TCP buffer sizes
net.ipv4.tcp_rmem = 4096 262144 8388608
net.ipv4.tcp_wmem = 4096 262144 8388608
```

  After these changes are made to the **/etc/sysctl.conf** file, apply the changes to your system:

  1. Issue the **sysctl -p /etc/sysctl.conf** command to set the kernel settings.
  2. Issue the **mmstartup -a** command to restart GPFS

Disk I/O

To optimize disk I/O performance, you should consider the following options for NSD servers or other GPFS nodes that are directly attached to a SAN over a Fibre Channel (FC) network.

1. The storage server cache settings can impact GPFS performance if not set correctly.
2. When the storage server disks are configured for RAID5, some configuration settings can affect GPFS performance. These settings include:
   - GPFS block size
   - Maximum I/O size of the Fibre Channel host bus adapter (HBA) device driver
   - Storage server RAID5 stripe size

   Note: For optimal performance, GPFS block size should be a multiple of the maximum I/O size of the FC HBA device driver. In addition, the maximum I/O size of the FC HBA device driver should be a multiple of the RAID5 stripe size.
3. These suggestions may avoid the performance penalty of read-modify-write at the storage server for GPFS writes. Examples of the suggested settings are:
   - 8+P RAID5
     - GPFS block size = 512K
     - Storage Server RAID5 segment size = 64K (RAID5 stripe size=512K)

- Maximum IO size of FC HBA device driver = 512K
- 4+P RAID5
  - GPFS block size = 256K
  - Storage Server RAID5 segment size = 64K (RAID5 stripe size = 256K)
  - Maximum IO size of FC HBA device driver = 256K

For the example settings using 8+P and 4+P RAID5, the RAID5 parity can be calculated from the data written and will avoid reading from disk to calculate the RAID5 parity. The maximum IO size of the FC HBA device driver can be verified using **iostat** or the Storage Server performance monitor. In some cases, the device driver may need to be patched to increase the default maximum IO size.

4. The GPFS parameter **maxMBpS** can limit the maximum throughput of an NSD server or a single GPFS node that is directly attached to the SAN with a FC HBA. The default value is 2048. The **maxMBpS** parameter is changed by issuing the **mmchconfig** command. If this value is changed, restart GPFS on the nodes, and test the read and write performance of a single node and a large number of nodes.

AIX configuration and tuning considerations

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

GPFS use with Oracle

When using GPFS with Oracle, configuration and tuning considerations include the following.

- When setting up your LUNs, it is important to create the NSD such that they map one to one with a LUN that is a single RAID device.
- For file systems holding large Oracle databases, set the GPFS file system block size through the **mmcrfs** command using the **-B** option, to a large value:
  - 512 KB is generally suggested.
  - 256 KB is suggested if there is activity other than Oracle using the file system and many small files exist which are not in the database.
  - 1 MB is suggested for file systems 100 TB or larger.

  The large block size makes the allocation of space for the databases manageable and has no affect on performance when Oracle is using the Asynchronous I/O (AIO) and Direct I/O (DIO) features of AIX.
- Set the GPFS worker threads through the **mmchconfig worker1Threads** command to allow the maximum parallelism of the Oracle AIO threads.
  - Adjust the GPFS prefetch threads accordingly through the **mmchconfig prefetchThreads** command. The maximum value of **prefetchThreads** plus **worker1Threads** plus **nsdMaxWorkerThreads** is 8192 on all 64-bit platforms.
  - When requiring GPFS sequential I/O, set the prefetch threads between 50 and 100 (the default is 72).

  Note: These changes through the **mmchconfig** command take effect upon restart of the GPFS daemon.
- The number of AIX AIO *kprocs* to create should be approximately the same as the GPFS **worker1Threads** setting.
- The AIX AIO *maxservers* setting is the number of *kprocs* PER CPU. It is suggested to set is slightly larger than the value of **worker1Threads** divided by the number of CPUs. For example if **worker1Threads** is set to 500 on a 32-way SMP, set *maxservers* to 20.
- Set the Oracle database block size equal to the LUN segment size or a multiple of the LUN pdisk segment size.
- Set the Oracle read-ahead value to prefetch one or two full GPFS blocks. For example, if your GPFS block size is 512 KB, set the Oracle blocks to either 32 or 64 16 KB blocks.

- Do not use the **dio** option on the **mount** command as this forces DIO when accessing *all* files. Oracle automatically uses DIO to open database files on GPFS.
- When running Oracle RAC 10g, it is suggested you increase the value for **OPROCD_DEFAULT_MARGIN** to at least 500 to avoid possible random reboots of nodes.

  In the control script for the Oracle CSS daemon, located in **/etc/init.cssd** the value for **OPROCD_DEFAULT_MARGIN** is set to 500 (milliseconds) on all UNIX derivatives except for AIX. For AIX this value is set to 100. From a GPFS perspective, even 500 milliseconds maybe too low in situations where node failover may take up to a minute or two to resolve. However, if during node failure the surviving node is already doing direct IO to the **oprocd** control file, it should have the necessary tokens and indirect block cached and should therefore not have to wait during failover.

Chapter 8. Steps to permanently uninstall GPFS and/or Protocols

GPFS maintains a number of files that contain configuration and file system related data. Since these files are critical for the proper functioning of GPFS and must be preserved across releases, they are not automatically removed when you uninstall GPFS.

Follow these steps if you do not intend to use GPFS on any of the nodes in your cluster and you want to remove all traces of GPFS:

Attention: After following these steps and manually removing the configuration and file system related information, you will permanently lose access to all of your current GPFS data.

1. Unmount all GPFS file systems on all nodes by issuing the **mmumount all -a** command.
2. Issue the **mmdelfs** command for each file system in the cluster to remove GPFS file systems.
3. Issue the **mmdelnsd** command for each NSD in the cluster to remove the NSD volume ID written on sector 2.

   If the NSD volume ID is not removed and the disk is again used with GPFS at a later time, you will receive an error message when issuing the **mmcrnsd** command. For more information, see *NSD creation fails with a message referring to an existing NSD* in *IBM Spectrum Scale: Problem Determination Guide*.
4. Issue the **mmshutdown -a** command to shutdown GPFS on all nodes.
5. Uninstall GPFS from each node:
   - For your Linux nodes, run the de-installation program to remove GPFS for the correct version of the packages for your hardware platform and Linux distribution. For example, on SLES and Red Hat Enterprise Linux nodes:

     `rpm -e gpfs.crypto` (IBM Spectrum Scale Advanced Edition only)

     `rpm -e gpfs.adv` (IBM Spectrum Scale Advanced Edition only)

     `rpm -e gpfs.ext` (IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition only)

     ```
     rpm -e gpfs.gpl
     rpm -e gpfs.msg.en_US
     rpm -e gpfs.base
     rpm -e gpfs.docs
     rpm -e gpfs.gskit
     ```

     on Debian Linux nodes:

     `dpkg -P gpfs.crypto` (IBM Spectrum Scale Advanced Edition only)

     `dpkg -P gpfs.adv` (IBM Spectrum Scale Advanced Edition only)

     `dpkg -P gpfs.ext` (IBM Spectrum Scale Standard Edition and IBM Spectrum Scale Advanced Edition only)

     ```
     dpkg -P gpfs.gpl
     dpkg -P gpfs.msg.en_US
     dpkg -P gpfs.base
     dpkg -P gpfs.docs
     dpkg -P gpfs.gskit
     ```
   - For your AIX nodes:

     ```
     installp -u gpfs
     ```

© Copyright IBM Corp. 2014, 2016

- For your Windows nodes, follow these steps:
  a. Open **Programs and Features** in the Control Panel.
  b. Uninstall **IBM General Parallel File System**.
  c. Reboot the system.
  d. From a Command Prompt, run the following command:
     sc.exe delete mmwinserv
6. Remove the **/var/mmfs** and **/usr/lpp/mmfs** directories.
7. Remove all files that start with **mm** from the **/var/adm/ras** directory.
8. Remove **/tmp/mmfs** directory and its content, if present.

Cleanup procedures required if reinstalling with the spectrumscale installation toolkit

Before you can reinstall with the **spectrumscale** installation toolkit, you must perform some cleanup procedures first.

You can use the following procedures to clean up various stages of the previous GPFS installation and protocols deployment.

Starting the cleanup process

1. Clean up the installer directory. To do so, issue the following command:
   ```
   mmdsh rm -rf /usr/lpp/mmfs/4.2.0.0
   ```
2. Check `/etc/resolv.conf` for any changes that pointed the authentication directory to DNS, and remove them.
3. If the object protocol was enabled and storage policies were created, use the **mmobj policy list -v** command and save the output list before you execute the **mmobj service disable OBJ** command.

Cleaning up the authentication configurations

Use these commands to completely remove the current authentication configuration for all protocols, NFS/SMB only, or Object only. Be aware that doing so may cause loss of access to previously written data.

1. Remove the configurations. To do so, issue the following commands:
   ```
   # mmuserauth service remove --data-access-method file
   # mmuserauth service remove --data-access-method object
   ```
2. Delete the ID mapping information when authentication was configured with AD. To do so, use the following command.

   Note: Once this ID mapping information is deleted, you might not be able to access existing files (or you might experience other unforeseen access issues), so do this only if you do not need access to existing data.
   ```
   # mmuserauth service remove --data-access-method file --idmapdelete
   ```

Disabling CES

Disable CES on every node *nodeNameX*. To do so, issue the following command:

```
mmchnode -N nodeNameX --ces-disable
```

Protocol cleanup steps

1. Use the following steps to clean up some of the object state on the system:

```
mmdsh -N cesNodes systemctl stop postgresql-obj
rm -rf path_to_cesSharedRoot/object/keystone
```

Delete the objectization temp directory. In the following example, the temp directory was created in filesystem `fs1` at `/ibm/fs1/ibmobjectizer/tmp`:

```
rm -rf  /ibm/fs1/ibmobjectizer/tmp
```

2. Remove the fileset created for object.

   In the following example, the configured fileset `object_fileset` in file system `fs1` was linked at `/gpfs/fs1`.

   ```
   mmlsfileset fs1
   mmunlinkfileset fs1 object_fileset
   mmdelfileset fs1 object_fileset -f
   ```

3. Remove any fileset created for an object storage policy.

   Run the **mmobj policy list -v** command. If for some reason **mmobj policy list -v** cannot be executed (For example, if the Object Protocol or CES was already disabled) or you cannot detect which filesets are used for object storage policies, contact the IBM Support Center.

   For example:

   If the **mmobj policy list** returned the following and the filesets got created in filesystem fs1:

```
Index       Name          Deprecated  Fileset          Fileset Path                       Functions               Function Details
-------------------------------------------------------------------------------------------------------------------------------------
0           SwiftDefault              object_fileset   /ibm/cesSharedRoot/object_fileset
11751509160 sof-policy1               obj_sof-policy1  /ibm/cesSharedRoot/obj_sof-policy1 file-and-object-access  regions="1"
11751509230 mysofpolicy               obj_mysofpolicy  /ibm/cesSharedRoot/obj_mysofpolicy file-and-object-access  regions="1"
11751510260 Test19                    obj_Test19       /ibm/cesSharedRoot/obj_Test19                              regions="1"
```

   Use the following commands for each row except the first:

   ```
   mmunlinkfileset fs1 sof-policy1
   mmdelfileset fs1 sof-policy1 -f
   ```

4. Clean up installer remnants. To do so, issue the following command:

   ```
   mmdsh rm -rf /root/.chef
   ```

5. Clear the **cesSharedRoot** configuration. To do so, issue the following command:

   ```
   mmchconfig cesSharedRoot=DEFAULT
   ```

6. Check your recent rpm installs to determine which rpms need to be removed. To do so, issue the following commands:

   ```
   rpm -qa --last|more
   ```

   Once you have determined which rpms need to be removed, do so by issuing the following commands, as appropriate:

   ```
   mmdsh yum erase gpfs.smb nfs-ganesha-mount nfs-ganesha-vfs PyQt4 nfs-ganesha-utils phonon-
   backend-gstreamer phonon sip qt-x11 kde-filesystem qt qt-settings libmng nfs-ganesha-proxy nfs-ganesha-nullfs nfs-
   ganesha-gpfs nfs-ganesha chef -y

   mmdsh yum erase python-webob MySQL-python mariadb-server perl-DBD-MySQL mariadb python-
   futures python-warlock python-jsonpatch python-jsonschema python-jsonpointer python-cmd2 python-
   cliff pyparsing memcached python-oslo-utils python-oslo-serialization python-oslo-i18n python-babel
   babel python-keyring python-stevedore python-prettytable python-pbr python-oslo-config python-
   netaddr python-iso8601 python-dnspython python-urllib3 python-six python-requests python-paste-
   deploy python-tempita python-paste python-netifaces python-simplejson python-greenlet python-
   eventlet -y

   mmdsh yum erase gpfs.smb gpfs.smb-debuginfo sssd-ipa sssd python-sssdconfig sssd-proxy sssd-ldap
   sssd-krb5 libipa_hbac sssd-krb5-common sssd-common-pac sssd-ad sssd-common cyrus-sasl-gssapi c-
   ares libsss_idmap libdhash yp-tools ypbind -y

   mmdsh yum erase python-routes python-repoze-lru python-oauthlib python-crypto python-qpid-
   common python-qpid python-dogpile-cache python-fixtures python-dogpile-core python-testtools
   python-extras python-oslo-context python-kombu python-anyjson python-amqp python-mimeparse
   python-markupsafe PyYAML python-passlib libyaml python-ibm-db-sa python-ibm-db python-sqlparse
   python-memcached python-webob python-posix_ipc python-sqlalchemy python-pbr python-greenlet -y

   mmdsh yum erase nfs-ganesha gpfs.gss.pmsensors gpfs.gss.pmcollector boost-regex -y
   mmdsh yum erase gpfs.smb python-oslo-i18n openstack-utils qt-settings nfs-ganesha-gpfs nfs-ganesha -y
   mmdsh yum erase python-ordereddict python-routes python-passlib python-amqp python-crypto -y
   ```

```
mmdsh yum erase mariadb-libs -y

mmdsh yum erase pmswift nfs-ganesha-gpfs nfs-ganesha gpfs.smb -y

mmdsh yum erase python-cryptography python-enum34 swift3 python-pyeclib liberasurecode openstack-
utils crudini python-msgpack python-wsgiref python-extras python-pycparser python-ply python-cffi
python-ibm-db openstack-selinux xmlsec1 python-pyasn1 python-mimeparse python-retrying
postgresql-server postgresql python-psycopg2 python-repoze-who jerasure gf-complete python-zope-
interface postgresql-libs pytz python-oslo-context python-webob python-six nfs-ganesha-gpfs nfs-
ganesha -y
mmdsh yum erase spectrum-scale-object spectrum-scale-object-selinux -y
```

7. Run **rpm -qa --last | more** on all nodes again, as the preceding list does not contain everything in the new builds.

   If `pmswift` rpm still exists, remove it as follows:

   ```
   mmdsh rpm -e pmswift-4.2.0-0.noarch –noscripts
   ```

   You might also need to remove `gpfs.smb`.

8. Clean up additional chef files. To do so, issue the following commands:

   ```
   mmdsh rm -rf /var/chef
   mmdsh rm -rf /etc/chef
   mmdsh rm -rf /opt/chef
   mmdsh rm -rf /root/.chef
   mmdsh rm -rf /root/.berkshelf
   ```

9. Clean up performance monitoring tool files. To do so, issue the following commands:

   ```
   mmdsh rm -rf /opt/IBM/zimon*
   mmdsh rm -rf /usr/IBM/zimon*
   mmdsh rm -rf /var/log/cnlog/zimon*
   mmdsh rm -rf /var/lib/yum/repose/x86_64/7Server/*zimon*
   mmdsh rm -rf /var/lib/yum/repose/ppc64/7Server/*zimon*
   ```

10. Clean up CTDB. To do so, issue the following commands:

    ```
    mmdsh rm -rf /var/lib/ctdb
    ```

11. Clean up swift files. To do so, issue the following commands:

    ```
    mmdsh rm -rf /etc/swift
    mmdsh rm -rf /var/lib/mysql
    mmdsh rm -rf /etc/keystone
    mmdsh rm -rf /var/lib/keystone
    mmdsh rm -rf /root/openrc
    mmdsh rm -rf /var/cache/yum/x86_64/7Server/*
    mmdsh rm -rf /var/cache/yum/ppc64/7Server/*
    mmdsh rm -rf /var/log/maria*
    mmdsh rm -rf /usr/share/pixmaps/comps/maria*
    mmdsh rm -rf /var/log/keystone
    mmdsh rm -rf /var/spool/cron/keystone
    mmdsh rm -rf /usr/lib/python2.7/site-packages/sos/plugins/openstack_keystone*
    mmdsh rm -rf /tmp/keystone-signing-swift
    mmdsh rm -rf /usr/lib/python2.7/site-packages/sos/plugins/*
    mmdsh rm -rf /var/lib/yum/repos/x86_64/7Server/icm_openstack
    mmdsh rm -f /usr/lib/python2.7/site-packages/swiftonfile-2.3.0_0-py2.7.egg
    mmdsh rm -f /usr/bin/*objectizer*.pyc
    mmdsh rm -f /usr/bin/generate_dbmap.pyc
    mmdsh systemctl disable openstack-keystone.service
    mmdsh systemctl disable openstack-swift-container-updater.service
    mmdsh systemctl disable openstack-swift-container-update
    mmdsh systemctl disable openstack-swift-object-updater.service
    mmdsh systemctl disable openstack-swift-container-auditor.service
    mmdsh systemctl disable openstack-swift-container-replicator.service
    mmdsh systemctl disable openstack-swift-container.service
    mmdsh systemctl disable openstack-swift-object-replicator.service
    mmdsh systemctl disable openstack-swift-object.service
    mmdsh systemctl disable openstack-keystone.service
    mmdsh systemctl disable openstack-swift-account-reaper.service
    mmdsh systemctl disable openstack-swift-account-auditor.service
    mmdsh systemctl disable openstack-swift-account-replicator.service
    mmdsh systemctl disable openstack-swift-account.service
    mmdsh systemctl disable openstack-swift-proxy.service
    mmdsh systemctl disable openstack-swift-object-auditor.service
    ```

```
mmdsh systemctl disable openstack-swift-object-expirer.service
mmdsh systemctl disable openstack-swift-container-reconciler.service
mmdsh rm -rf
/var/lib/yum/yumdb/o/0b01eb65826df92befd8c161798cb842fa3c941e-openstack-utils-2015.1-201502031913.ibm.el7.7-noarch
```

12. Reboot. To do so, issue the following command:

    ```
    mmdsh shutdown -r now
    ```

    If some nodes do not fully come up, do the following:

    a. Power off that node.

    b. Wait one minute, then power the node back on.

    Note: Some nodes might need to be powered off and on more than once.

13. Clean up Yum on all nodes. To do so, issue the following commands:

    ```
    mmdsh yum clean all
    mmdsh rm -rf /etc/yum.repos.d/gpfs.repo /etc/yum.repos.d/icm* /etc/yum.repos.d/ces.repo
    mmdsh rm -rf /etc/yum.repos.d/epel* /etc/yum.repos.d/rdo*
    mmdsh rm -rf /var/lib/yum/repos/x86_64/7Server/*
    mmdsh rm -rf /var/lib/yum/repos/ppc64/7Server/*
    ```

14. Remove GPFS. For more information see "Procedure for installing GPFS on Windows nodes" on page 164.

    Notes:

    - This step is not required if you know that GPFS has not changed between old and new builds.
    - This step is not required if you prefer to perform an upgrade of GPFS.
    - This step is not required if you would like to keep the base GPFS installation intact and merely rerun the protocols deployment.
    - (To permanently remove GPFS, see Chapter 8, "Steps to permanently uninstall GPFS and/or Protocols," on page 193.)

    a. Check which GPFS rpms are on each node. To do so, issue the following command:

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

    The system displays output similar to the following:

    ```
    rpm -qa|grep gpfs
    gpfs.ext-4.2.0-0.x86_64
    gpfs.msg.en_US-4.2.0-0.noarch
    gpfs.gskit-8.0.50-47.x86_64
    gpfs.crypto-4.2.0-0.x86_64
    gpfs.adv-4.2.0-0.x86_64
    gpfs.docs-4.2.0-0.noarch
    gpfs.base-4.2.0-0.x86_64
    gpfs.gpl-4.2.0-0.noarch
    ```

    b. Before removing anything, make sure that GPFS is shut down on all nodes. To do so, issue the following command:

    ```
    mmshutdown -a
    ```

    c. Remove the rpms. To do so, issue the following commands *in the order shown*.

    Note: When you remove `gpfs.base`, you will lose **mmdsh** access.
    Therefore, be sure to remove `gpfs.base` last, as shown here.

    ```
    mmdsh rpm -e gpfs.base-debuginfo-4.2.0-0.x86_64
    mmdsh rpm -e gpfs.crypto-4.2.0-0.x86_64
    mmdsh rpm -e gpfs.adv-4.2.0-0.x86_64
    mmdsh rpm -e gpfs.ext-4.2.0-0.x86_64
    mmdsh rpm -e gpfs.msg.en_US-4.2.0-0.noarch
    mmdsh rpm -e gpfs.gskit-8.0.50-47.x86_64
    mmdsh rpm -e gpfs.docs-4.2.0-0.noarch
    mmdsh rpm -e gpfs.gpl-4.2.0-0.noarch
    mmdsh rpm -e gpfs.base-4.2.0-0.x86_64
    ```

15. Reinstall GPFS.
16. Proceed to "Installation prerequisites" on page 109 and "Using the **spectrumscale** installation toolkit to perform installation tasks: Explanations and examples" on page 114.

Note: If you wish to remove all cluster configurations, you can also apply the **mmdelnode -f** command to each node; however, if you choose to do so, you will also have to remake `cluster/nsds/filesystems`.

Uninstalling the Performance Monitoring tool

You can uninstall the Performance Monitoring tool by running the following commands on all nodes that has monitoring enabled on it.

To uninstall the sensor on the node, use the **rpm -evh gpfs.gss.pmsensors** command.

To uninstall the collector on the node, use the **rpm -evh gpfs.gss.pmcollector** command.

To uninstall the Object proxy on the node, use the **rpm -evh pmswift** command.

For reinstalling sensors and the collector, see "Manually installing the Performance Monitoring tool" on page 139.

Uninstalling the IBM Spectrum Scale management GUI

Do the following to uninstall management GUI and remove the performance monitoring components that are installed for the GUI:

1. Issue the **systemctl stop** command as shown in the following example:

   ```
   systemctl stop gpfsgui
   psql postgres postgres -c "drop schema fscc cascade"
   ```

2. Issue the following commands to remove the performance monitoring components of the GUI:

   ```
   systemctl stop pmcollector
   rpm -e gpfs.gui* gpfs.gss.pmsensors*  gpfs.gss.collector*
   ```

3. Issue the following commands on all nodes of the cluster to remove the performance monitoring components installed on them:

   ```
   systemctl stop pmsensors
   rpm -e gpfs.gss.pmsensors*
   ```

Removing nodes from management GUI-related node class

If you are reinstalling IBM Spectrum Scale, in some scenarios you might need to remove some nodes from the node classes that are used by the management GUI.

For information about these node classes, see "Node classes used for the management GUI" on page 149. If you want to remove the GUI designation of a node, you must remove it from the `GUI_MGT_SERVERS` node class. After you remove a node from this node class, it is no longer designated as a GUI node and GUI services are not started on this node after reinstalling IBM Spectrum Scale. To remove a node from the `GUI_MGT_SERVERS` node class, use the **mmchnodeclass** command only when the GUI services are not running.

On the node that you want to remove from the `GUI_MGT_SERVERS` node class, run the following commands:

```
systemctl stop gpfsgui
mmchnodeclass GUI_MGT_SERVERS delete -N guinode
```

These commands stop the GUI services and remove the GUI node from the `GUI_MGT_SERVERS` node class.

| **Note:** If you use **mmchnodeclass** to change the GUI_MGT_SERVERS node class while the GUI services are
| running, the management GUI adds the removed node to the GUI_MGT_SERVERS node class again.

Chapter 9. Shutting down an IBM Spectrum Scale cluster

Use the following information to shut down an IBM Spectrum Scale cluster in an emergency situation.

1. Stop the protocol services on all protocol nodes in the cluster using the **mmces service stop** command. For example:

   ```
   mmces service stop nfs -a
   mmces service stop smb -a
   mmces service stop obj -a
   ```

2. Unmount all file systems, except the CES shared root file system, on all nodes in the cluster using the **mmumount** command.
3. Stop GPFS daemons on all protocol nodes in the cluster using the **mmshutdown -N cesNodes** command.
4. Unmount all file systems on all nodes in the cluster using the **mmumount all -a** command.
5. Stop GPFS daemons on all nodes in the cluster using the **mmshutdown -a** command.

After performing these steps, depending on your operating system, shut down your servers accordingly.

Before shutting down and powering up your servers, consider the following:

- You must shut down NSD servers before the storage subsystem. While powering up, the storage subsystem must be online before NSD servers are up so that LUNs are visible to them.
- In a power-on scenario, verify that all network and storage subsystems are fully operational before bringing up any IBM Spectrum Scale nodes.
- On the Power platform, you must shut down operating systems for LPARs first and then power off servers using the HMC. The HMC must be the last to be shut down and the first to be powered up.
- It's preferable to shut down your Ethernet and InfiniBand switches using the management console instead of powering them off. In any case, network infrastructure such as switches or extenders must be powered off last.
- After starting up again, verify that functions such as AFM, policies, etc. are operational. You might need to manually restart some functions.
- There are a number other GPFS functions that could be interrupted by a shutdown. Ensure that you understand what else might need to be verified depending on your environment.

© Copyright IBM Corp. 2014, 2016

Chapter 10. Considerations for GPFS applications

Application design should take into consideration the exceptions to Open Group technical standards with regard to the **stat()** system call and NFS V4 ACLs. Also, a technique to determine whether a file system is controlled by GPFS has been provided.

For more information, see the following topics in *IBM Spectrum Scale: Administration and Programming Reference*:

- *Exceptions to Open Group technical standards*
- *Determining if a file system is controlled by GPFS*
- *GPFS exceptions and limitation to NFS V4 ACLs*

© Copyright IBM Corp. 2014, 2016

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.

© Copyright IBM Corporation © IBM 2014, 2016

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.

© Copyright IBM Corporation © IBM 2014, 2016

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. H6MA/Building 707
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml.

Intel is a trademark of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of the Open Group in the United States and other countries.

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 MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

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.

© Copyright IBM Corporation © IBM 2014, 2016

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

Special characters

A

B

C

© Copyright IBM Corp. 2014, 2016

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Product Number: 5725-Q01
5641-GPF
5725-S28

Printed in USA

GA76-0441-06