TSM for Space Management for UNIX – GPFS Integration

Part I: Policy-driven Threshold Migration

Dominic Mueller-Wicke and Christian Mueller

Version 1.2

Copyright Notice

Trademarks

IBM, the IBM logo, Tivoli, the Tivoli logo, AIX, Cross-Site, NetView, OS/2, Planet Tivoli, RS/6000, Tivoli Certified, Tivoli Enterprise, Tivoli Enterprise Console, Tivoli Ready, and TME are trademarks or registered trademarks of International Business Machines Corporation or Tivoli Systems Inc. in the United States, other countries, or both.

Lotus is a registered trademark of Lotus Development Corporation.
Microsoft, Windows, Windows NT, and the Windows logo 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.
C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.
PC Direct is a trademark of Ziff Communications Company in the United States, other countries, or both and is used by IBM Corporation under license.
ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks of Intel Corporation in the United States, other countries, or both. For a complete list of Intel trademarks, see http://www.intel.com/sites/corporate/trademarx.htm.
SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC. For further information, see http://www.setco.org/aboutmark.html.
Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Notices

References in this publication to Tivoli Systems or IBM products, programs, or services do not imply that they will be available in all countries in which Tivoli Systems or IBM operates. Any reference to these products, programs, or services is not intended to imply that only Tivoli Systems or IBM products, programs, or services can be used. Subject to valid intellectual property or other legally protectable right of Tivoli Systems or IBM, any functionally equivalent product, program, or service can be used instead of the referenced product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by Tivoli Systems or IBM, are the responsibility of the user. Tivoli Systems or IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785,

U.S.A

About the Tivoli Field Guides

Sponsor

Tivoli Customer Support sponsors the Tivoli Field Guide program.

Authors

Those who write field guides belong to one of these three groups:

- Tivoli Support and Services Engineers who work directly with customers
- Tivoli Customers and Business Partners who have experience using Tivoli software in a production environment
- Tivoli developers, testers, and architects

Audience

The field guides are written for all customers, both new and existing. They are applicable to external audiences including executives, project leads, technical leads, team members, and to internal audiences as well.

Types of Field Guides

Two types of Tivoli Field Guides describe how Tivoli products work and how they are used in real life situations:

- Field Guides for technical issues are designed to address specific technical scenarios or concepts that are often complex to implement or difficult to understand, for example: endpoint mobility, migration, and heartbeat monitoring.
- Field Guides for business issues are designed to address specific business practices that have a high impact on the success or failure of an ESM project, for example: change management, asset Management, and deployment phases.

Purposes

The Field Guide program has two major purposes:

- To empower customers & business partners to succeed with Tivoli software by documenting and sharing product information that provides accurate and timely information on Tivoli products and the business issues that impact an enterprise systems management project
- To leverage the internal knowledge within Tivoli Customer Support and Services and the external knowledge of Tivoli customers and Business Partners

Availability

All completed field guides are available free to registered customers and internal IBM employees at the following Web site:

http://www.ibm.com/software/sysmgmt/products/support/Field_Guides.html

Authors can submit proposals and access papers by e-mail:

mailto:Tivoli_eSupport_Feedback@us.ibm.com

Introduction

The Article series covers the integrated functionality of the IBM products GPFS and TSM for Space Management (HSM). This first part deals with the GPFS policy based threshold migration in detail with all involved components on the GPFS and the TSM HSM side as well as the setup and configuration of such an environment for the different GPFS and TSM for Space Management versions.

GPFS provides a disk-based Information Lifecycle Management (ILM) implementation with the storage pool concept. The Tivoli Storage Manager for Space Management Client (HSM) integrates tape into the GPFS ILM solution. With the functionality explained in this article, the HSM client can be used with the GPFS policy-based storage management features to do ILM to tape.

The explained approach of a policy-based threshold migration can be used in very large storage installations, instead of the classical HSM-controlled threshold migration to be able to handle up to billions of files and scale out with the number of nodes added to a GPFS cluster.

IBM TSM for Space Management (HSM) for UNIX

TSM for Space Management is a Hierarchical Storage Management solution (a.k.a. HSM).
A HSM solution typically moves the files data to backend storage and just leaves a small stub file back in the local storage. The stub file consumes just minimal space but leaves all meta-data information on the local storage in such a way that for a user or a program the file looks like a normal local stored file. When the user or a program accesses the file, the HSM solution automatically recalls (moves back) the file`s data from the back-end storage and gives the reading application access to the file when all the data is back online. The back-end storage for TSM HSM is the TSM Server which handles the tier2 (disk storage) and tier3 (tape storage) of the storage hierarchy. That means TSM HSM virtually extends the managed file system with the space provided by the TSM server. Migrating files to the TSM server frees space for new data on your local file system and takes advantage of lower-cost storage resources that are available in your network environment.
The whole functionality to perform the virtual file space expansion is implemented in the product itself. In GPFS environments, it is recommended to use the GPFS - HSM integration explained in this article because of scalability and performance reasons.

The whole documentation about the TSM for Space Management product can be found online using the following URL:

https://www.ibm.com/developerworks/wikis/display/tivolistoragemanager/Home

IBM General Parallel File System (GPFS)

GPFS is the IBM high performance parallel file system which is basically a cluster file system which has the capability of concurrent read and write access to the same files. The entire file system is striped across all disks in one pool.

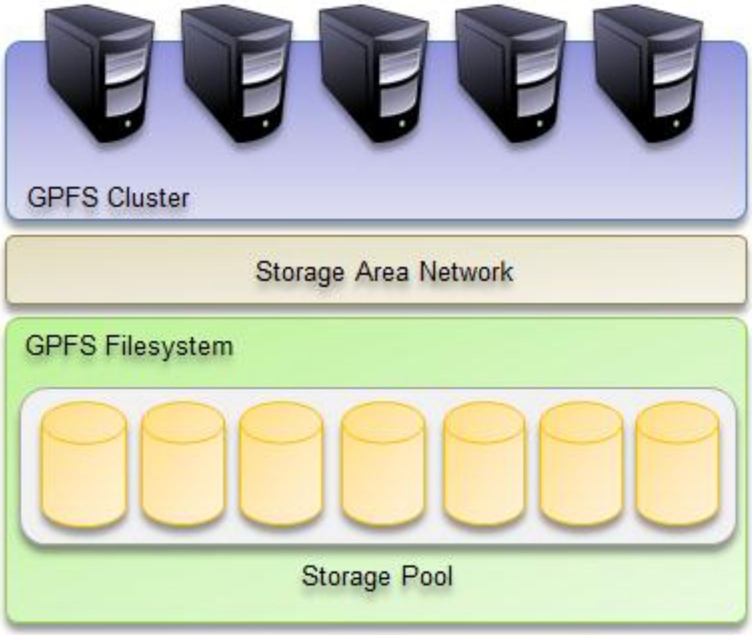


Figure 1: GPFS file system with a single storage pool

With version 3.1 GPFS has introduced the policy-based data management that automates the management of storage resources and the data stored on those resources. Policy-based data management is based on the storage pool concept. A storage pool is a collection of disks or RAIDs with similar properties that are managed together as a group. The group under which the storage pools are managed together is the file system.

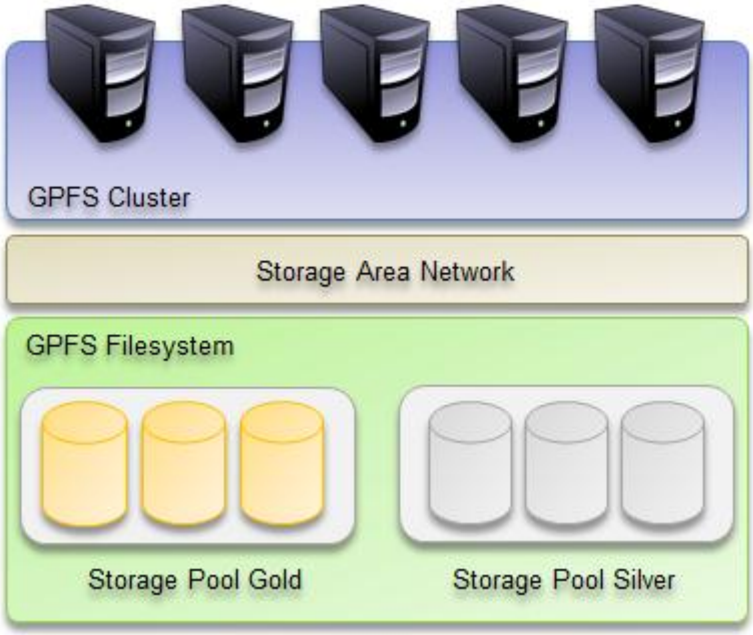


Figure 2: GPFS file system with two storage pools

GPFS provides a single name space across all pools. Files in the same directory can be in different pools. Files are placed in storage pools at creation time using placement policies. Files can be moved between pools based on migration policies and files can be removed based on given policies.

GPFS implements an engine for fast file system scans and file-system monitoring which is also known as policy engine. The policy engine implements a user programmable interface which is called the GPFS policy rule language. The article focuses on the usage of this engine in conjunction with the TSM for Space Management client.

A complete explanation of the SQL like policy rule language can be found in the GPFS documentation 'GPFS Advanced Administration Guide'. This documentation is available online using the following URL:

http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp

GPFS driven migration

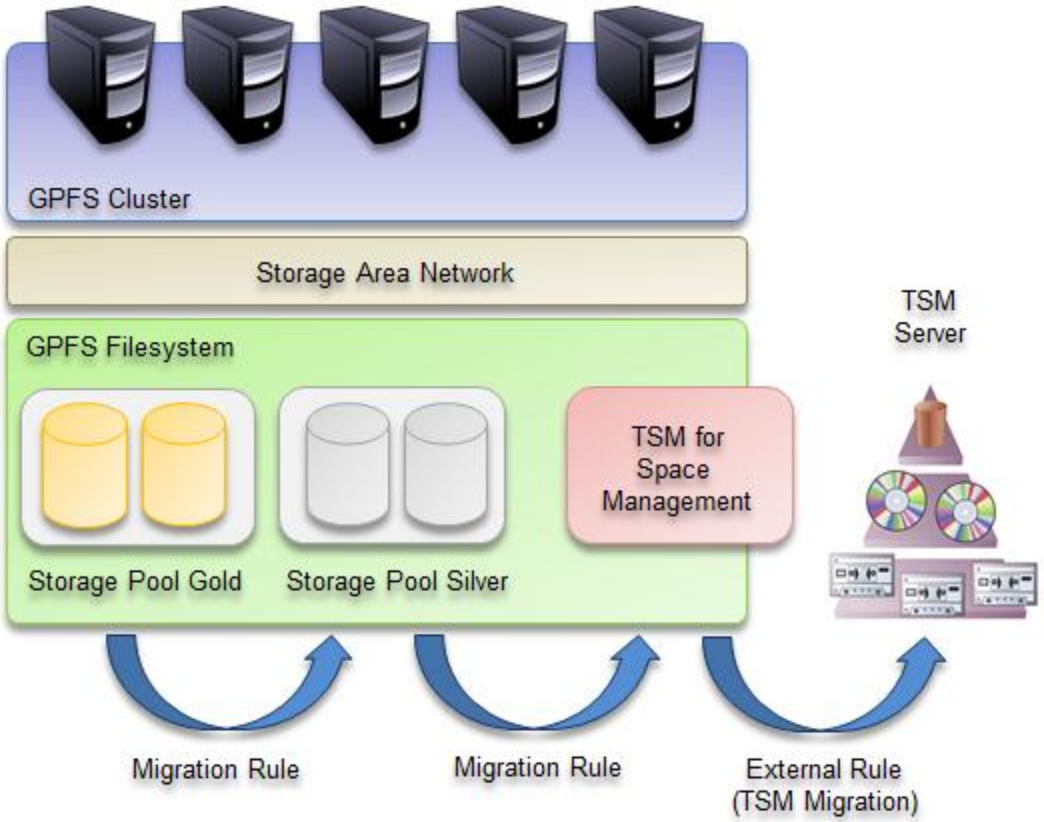


Figure 3: GPFS file system with an external HSM storage pool

GPFS 3.2 introduces a new feature called external storage pools. You can set up external storage pools and GPFS policies allowing the GPFS policy manager to coordinate file migrations from a native GPFS online pool to external pools on the Tivoli Storage Manager server. The GPFS policy manager invokes the migration through the HSM client command line interface.

The migration candidate selection is identical to the GPFS native pool to pool migration rule. The Policy Engine uses scripts to call the TSM command dsmmigrate -filelist for the migration of files from a native storage pool to the TSM server.

There are two different approaches that can be used to drive a HSM migration via GPFS policies. The two approaches are only different in how the mmapplypolicy command which does the policy scan is started.

Manual GPFS driven migration

The manual GPFS driven migration is performed when the user or a Unix cron job executes the mmapplypolicy command with a pre-defined migration policy. The following is an example of what such a migration policy can look like:

```
/* Define hsm storage manager as an external pool
   The OPTS '-v' is for debugging and causes the exec script to print
   additional information */
RULE EXTERNAL POOL 'hsm' EXEC '/var/mmfs/etc/mmpolicyExec-hsm.sample'
OPTS '-v'

/* Migrate data files to tape. */
RULE 'MigrateData' MIGRATE
      FOR FILESET('data')
      FROM POOL 'system'
            WEIGHT( (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME))
                    + (KB_ALLOCATED / 1024) )
      TO POOL 'hsm'
            WHERE KB_ALLOCATED > 0
```

The rule covers the migration of files from the system pool to the external HSM pool. This means that the data is physically moved to the TSM disk or tape pool which has to be defined in TSM.

Based on this rule the policy engine will collect all files that are stored in the storage pool named “system” and migrate these files to a temporary file list. The weight clause determines the order for migration. The goal is to migrate large, unused files. The weight is simply the number of days since the last access plus the file's size in megabytes. Thus a 1 day old file that is 10 MB in size has the same priority for migration as an 11 day old file that is 1MB in size. The where clause selects the files to be migrated. We need to eliminate the files that have already been migrated. In this example, it was assumed that all data blocks for migrated files are de-allocated.

If that policy was set via *the mmchpolicy /yourFS yourPolicyFileName* command the next execution of the *mmapplypolicy* command will use this policy to match it against the meta-data stored in the inode. Another way is to explicitly provide the rules file to *mmapplypolicy* like *mmapplypolicy Device –P yourPolicyFileName*. That means the *mmapplypolicy* command starts an inode scan in ascending order of the inode number and matches the meta-data for each inode with the rules in the policy file. If the rule matches the inode information the file is a candidate for migration and is stored in a temporary file list. When the inode scan has finished an EXEC script is invoked. In the example above the EXEC script is “/var/mmfs/etc/mmpolicyExec-hsm.sample”. This script performs the migration of the files via the *dsmmigrate –filelist* command.

Automatic GPFS threshold migration

The GPFS threshold migration is performed when the user has specified a threshold policy and the GPFS policy daemon is enabled to monitor the storage pools in the file system for that given threshold. If a pre-defined high threshold is reached which means the filling level of the storage pool reached the pre-defined high water mark the monitor daemon automatically starts the *mmapplypolicy* command to perform an inode scan. The following is an example how such a threshold policy looks like:

```
/* Define hsm storage manager as an external pool
   The OPTS '-v' is for debugging and causes the exec script to print
   additional information */
RULE EXTERNAL POOL 'hsm' EXEC '/var/mmfs/etc/mmpolicyExec-hsm.sample'
OPTS '-v'

RULE 'MigrateData' MIGRATE
      FOR FILESET('data')
      FROM POOL 'system'
            THRESHOLD(80,70)
                WEIGHT( (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME))
                    + (KB_ALLOCATED / 1024) )
      TO POOL 'hsm'
            WHERE KB_ALLOCATED > 0
```

The policy rule looks mostly like the one from the previous example. The difference is that it has a THRESHOLD clause. The threshold clause says “start migrating when the system pool is more than 80% full and continue migrating until the pool is less than 70% full.” The other parts of the rule and the procedure works like described above.

HSM automatic threshold migration versus GPFS driven threshold migration

The following section compares the classic HSM automigration approach with the GPFS driven threshold migration.

TSM for Space Management threshold migration concept:

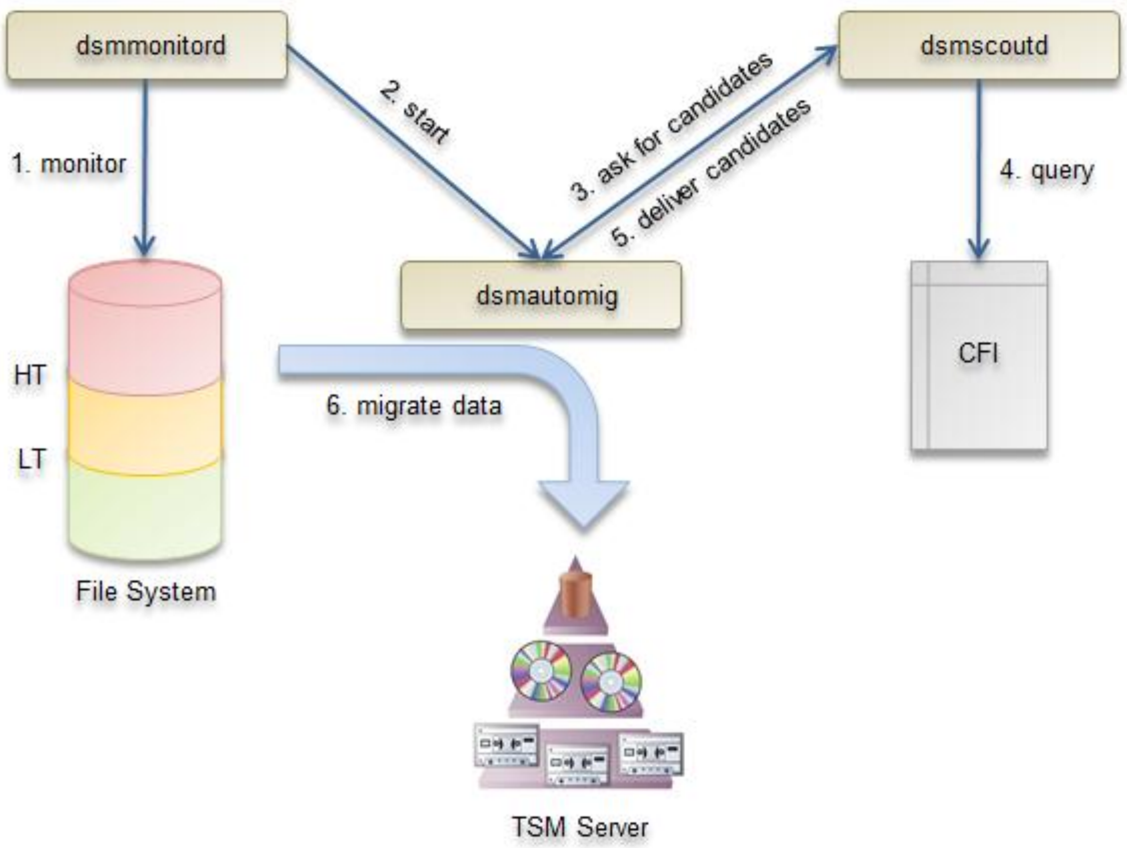


Figure 4. HSM classic automigration procedure

1. The HSM monitor daemon monitors the predefined thresholds of the managed file system.
2. When the file system filling level reaches the HT mark, the automigration process is started.
3. The dsmautomig asks the Scout daemon for candidates.
4. The Scout daemon’s internal search engine queries CFI for best candidates, pre-migrated first and then biggest and oldest.
5. The scout daemon delivers migration candidates to the automigration process.
6. The dsmautomig uses delivered candidates for migration to the TSM server.

Pro's:

- Whole process is controlled by HSM.
- Best candidates are selected to free up space efficiently.
- Works immediately after installation. No additional setup and configuration is needed.

Con's:

- The migration candidates are selected only by age and size file attributes.
- HSM needs to create and maintain a CFI (complete file index) which consumes up to 10% of the space in the HSM managed FS.
- Runs only from one node in the cluster and does not scale with the number of nodes where HSM is installed.
- The classical HSM approach scales up to 20 mio objects in a single file system. In GPFS environments it is recommended to use the GPFS driven threshold migration for file systems with more than 10 mio objects.

GPFS driven threshold migration concept:

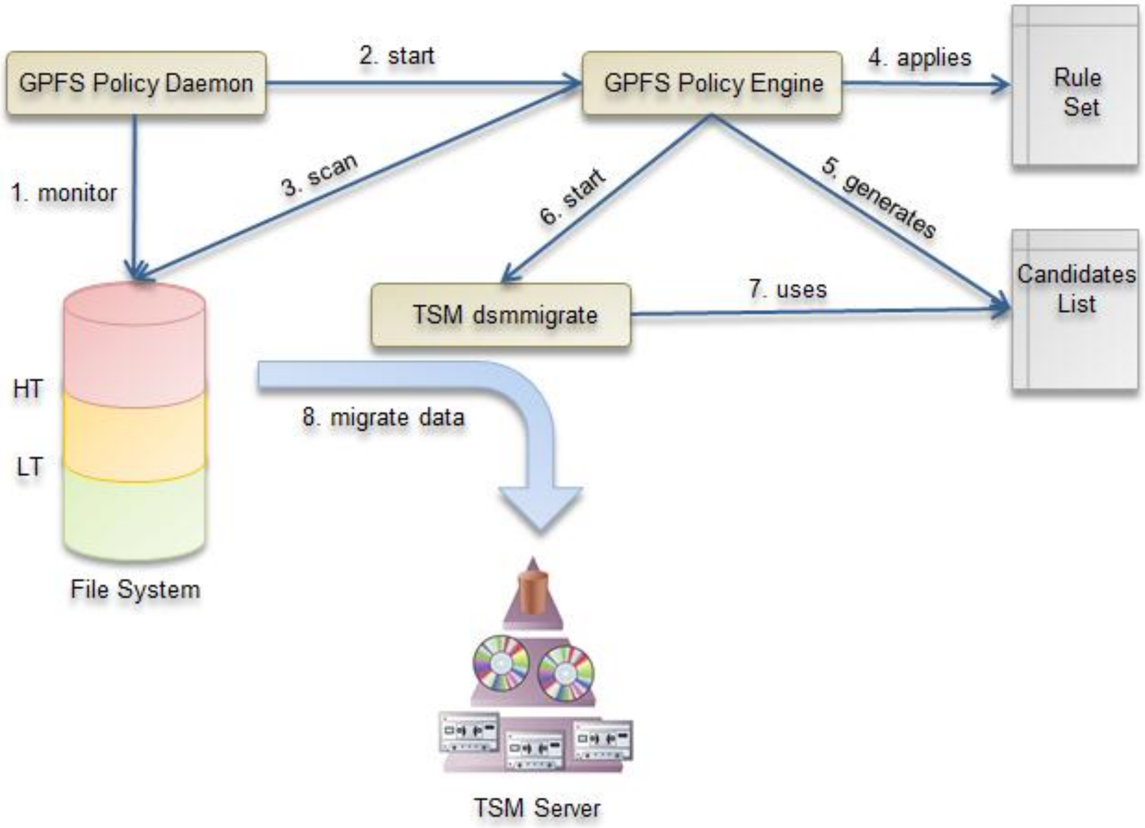


Figure 5. GPFS driven threshold migration procedure

- The GPFS policy daemon monitors HT/LT based on enabled policy.
- When the filling level reaches the HT mark the policy daemon starts the policy engine.

- The policy engine scans file system and generates a candidate list based on the enabled rule set for external migration (step 3, 4 and 5).
- The HSM command line tool dsmmigrate is called and migrates all files in the candidate list to the TSM server.

Pro's:

- Very fast candidate selection performed by the policy engine.
- All file attributes can be used and combined with a very powerful policy language to select migration candidates.
- Migration throughput scales out with the number of nodes where HSM is installed in the cluster.
- It is possible to handle billions of files in a single file system

Con's:

- More setup and configuration steps needed.
- If an insufficient migration policy is enabled an out-of-space condition could be created since not enough migration candidates can be found based on the given policy rule.

Multi node GPFS driven migration

The previously described GPFS driven migration approach can be distributed over several nodes in the cluster. HSM can scale out with the number of nodes added to the cluster. By distributing the migrations to several nodes the overall migration throughput can be increased.

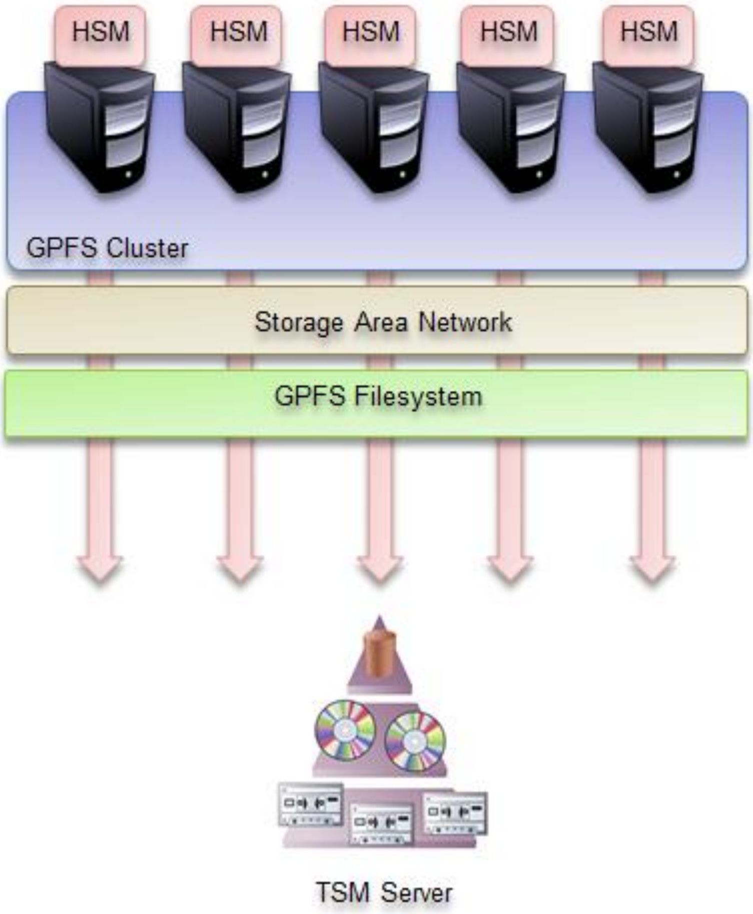


Figure 6. Multi-node GPFS driven migration

The GPFS Policy Engine runs simultaneously on all nodes where the HSM client is installed and on each node a migration candidate list is generated. The candidate lists are passed to the HSM command dsmmigrate to start migration from several nodes in parallel. In addition several HSM migrations are started in parallel on each node.

By default the GPFS policy engine generates migration candidates lists with 100 entries and starts 24 migration processes on each node. Each HSM migration processes one of these file lists with 100 migration candidates. The default settings for the number of concurrent migration processes and the number of files in the file lists can be configured via a GPFS command.

The same approach can be used for recalling files from the TSM server to a native GPFS storage pool. The Policy Engine uses script to call the dsmrecall -filelist command. Transparent recall by accessing the migrated file is still present. The manual recall of files can be significant improved by using the TSM for Space Management function “Tape Optimized Recall” which was introduced with TSM version 6.3. The whole documentation about the TSM for Space Management product can be found online using the following URL:

https://www.ibm.com/developerworks/wikis/display/tivolistoragemanager/Home

Setup a GPFS driven threshold migration environment

Prepare the HSM installation

GPFS provides the policy engine which is a GPFS specific implementation for fast file system scans. It is recommended to use this mechanism for HSM activities in GPFS environments. A special install mode can be used for HSM to install the modules which are needed in such an environment. Ensure the following export prior the installation of HSM:

```
export HSMINSTALLMODE=SCOUTFREE
```

NOTE: Manual setup. On AIX the HSMINSTALLMODE functionality wasn't implemented with TSM version 6.2. The configuration must be modified manually. If the HSM client is already installed on Linux without the HSMINSTALLMODE setting the manual setup has to be used also.

The following steps are required : modify the script

```
/[usr|opt]/tivoli/tsm/client/hsm/bin/rc.gpfshsm
```

remove the lines between #BEGINSCOUTFREE and #ENDSCOUTFREE.

```
#BEGINSCOUTFREE

if [ -x $bindir/dsmmonitord ]
then
   $bindir/dsmmonitord &
fi

if [ -x $bindir/dsmscoutd ]
then
   $bindir/dsmscoutd &
fi
#ENDSCOUTFREE
```

Add the following entry to the TSM options file

```
/[opt|usr]/tivoli/tsm/client/ba/bin/dsm.opt:
```

HSMDISABLEAUTOMIGDAEMONS YES

Stop the HSM daemons by using the command dsmmigfs stop. Send a SIGTERM to the process dsmwatchd. The daemon will be restarted from the /etc/inittab. After it's restart use the command dsmmigfs start to startup the HSM daemons.

This will lead to the same result as the INSTALLMODE setting described above. The HSM components scout daemon and monitor daemon will not started since these components are not needed in a GPFS driven environment.

Install the required components

Setup the TSM for Space Management client. The whole documentation about the TSM for Space Management product can be found online using the following URL:

https://www.ibm.com/developerworks/wikis/display/tivolistoragemanager/Home

After the installation the following components of HSM should run.

dsmwatchd – local and global failover mechanism

dsmrootd – HSM support for none root user

dsmrecalld (3 processes initial) – transparent recall of migrated files

Modify the TSM options file

/[opt|usr]/tivoli/tsm/client/ba/bin/dsm.opt.

Add the following entries:

```
HSMGROUPEDMIGRATE YES
```

It enables the HSM functionality to couple multiple files in one TSM server transaction. The client option is bounded by the TSM options TXNBYTELIMIT and TXNGROUPMAX. The TXNBYTELIMIT client option specifies the maximum transaction size for data transferred between the client and server. The range of values is from 300 KB to 32 GB. The default is 25,600 KB. The TXNGROUPMAX option specifies the number of objects that are transferred as a group between a client and the server between transactions commit points. The minimum value is 4 objects and the maximum value is 65000 objects. The default value is 4096 objects. The objects transferred are actual files, directories, or both. The server counts each file or directory as one object.

```
HSMEXTOBJIDATTR YES
```

This option is required to use the GPFS based file list reconciliation method. After the option was set each file migrated or premigrated will get an additional dmapi attribute.

HSM grouped migration

Since version 6.2.2 the Tivoli Storage Manager for Space Management allows to migrate multiple files per transaction to the TSM server. This significantly improves the migration performance when migrate a huge amount of small or medium size files (up to 100 MB) directly to tape. Such a bulk migration can be triggered by the GPFS driven migration via a file list. This is due to the fact that each file migrated to tape needs a tape commit which takes about 2 seconds. This amounts to 98% of the total runtime for medium and small size files, and reduces the migration throughput. With the option HSMGROUPEDMIGRATE set to YES files migrated to tape are grouped thus tape commit becomes a much less significant factor.

The default behavior is to transfer each file in a single transaction and make a stub file when the transaction was successful. When HSMGROUPEDMIGRATE is set to YES the transaction queue is automatically flushed to the server when the transaction byte limit or the transaction group limit is reached.

The transaction is also automatically flushed when a file is inserted into the transaction queue which has a different management class setting on the TSM server.

The HSM transaction process is performed in several steps. First step is to push a file that needs to be migrated to the transaction queue. In a second step the queue is flushed to server when the byte our group limit is reached and all the server responses are collected. When all files in the transaction are transferred successfully to the server and server has confirmed that the transaction was committed all files previously send are stubbed by HSM. When we get an error for a file in the transaction from the client side the file is skipped and all other files are send in one transaction. When we get an error during the transaction handling with the server the following happens based on the abort reason:

- When the server abort reason was "Waiting for Offline Media" then the entire transaction is resend.
- Otherwise each item in the transaction is resend in an individual transaction.

HSM cluster node configuration

The following is an explanation of the different manner the term node is used within a HSM installation on a GPFS cluster.

- Cluster node: the compute node which is a part of the AIX/Linux GPFS cluster
- Session node: (HSM definition “owner node”) this node manages the file system. That means the DMAPI dispositions are set on this node (The HSM node definition can change due to a failover. In addition a node can play different roles for different managed file systems).
- Source node: the node which has access to the GPFS file system, but does actual not manage the file system. This node can become an owner node in case of a failover. In addition this node will attend in the distributed recall.
- GPFS nodeset: the set of nodes having access to the same GPFS filesystem

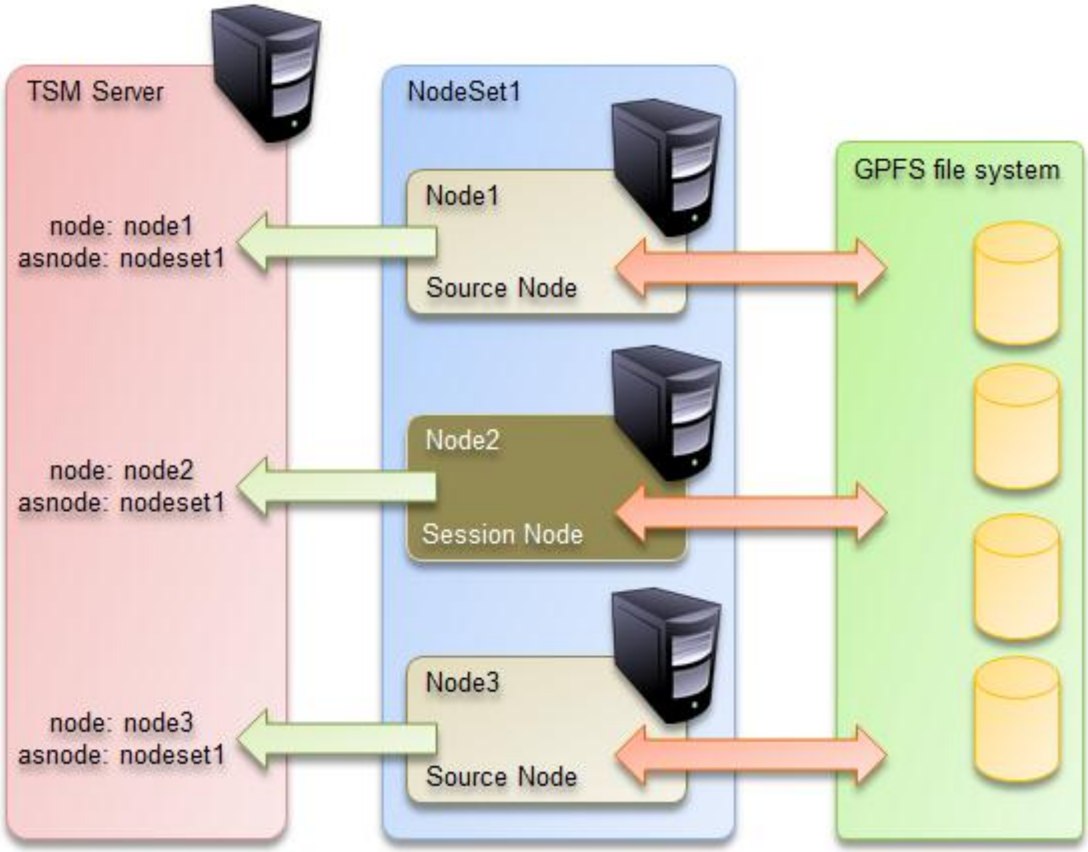


Figure 7. HSM – GPFS cluster configuration

To be able to migrate and recall files from every node in the cluster where HSM is installed (owner node and source nodes) it is necessary to use the ASNODENAME option to access a common file space on the TSM server from different source nodes and the owner node.

To use this option do the following:

1. Register each source node and the session node (where HSM is installed) with the Tivoli Storage Manager server (via the dsmadmc command).

   ```
   register node <NODENAME> passwd
   ```

2. The Tivoli Storage Manager server must grant proxy authority to the source nodes to access the node name where the GPFS file systems are stored, using the grant proxynode command (via the dsmadmc command).

   ```
   register node <CLUSTER_ASNODE> passwd archdel=yes
   backdel=yesgrant proxy target=<CLUSTER_ASNODE> agent=<NODENAME>
   ```

3. Update the system options file (dsm.sys) on each source node by specifying the asnodename option to access the common filespace for the HSM managed file systems on the Tivoli Storage Manager server.

   ```
   ASNODENAME <CLUSTER_ASNODE>
   ```

Verify the GPFS and TSM environment

Performing both hierarchical space management and backup using the GPFS policy engine is an advanced solution for TSM. Before starting with configuration, it is important to ensure that the core functionality of TSM and GPFS is given on the cluster. If this is the case, the cause for issues and malfunctions of the integrated functionality can by evaluated more easily.

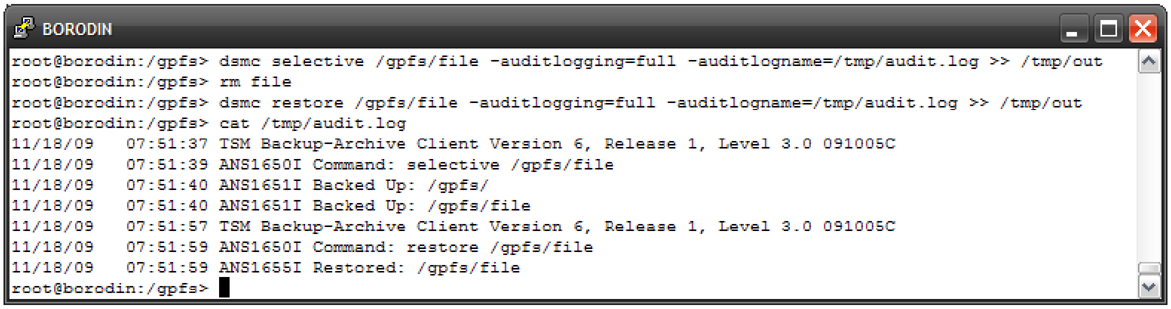
That means the cluster should run in a proper state and the TSM client should be able to backup/restore and migrate/recall files to the TSM server.

Check GPFS cluster state

Use the mmgetstate command to verify the GPFS cluster state. All cluster nodes should be “active.”

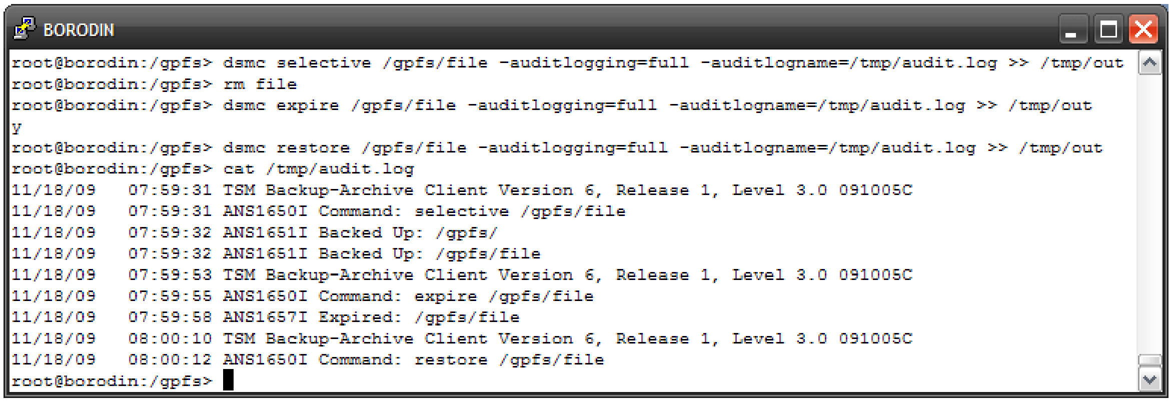
Check TSM Client functionality

On TSM, it is important to verify both clients: TSM for Space Management and Backup/Archive (BA). The BA client should be able to backup and restore a normal file. Furthermore, it should be possible to expire a backup copy of a file on the TSM server. Use the following command sequence for verification:



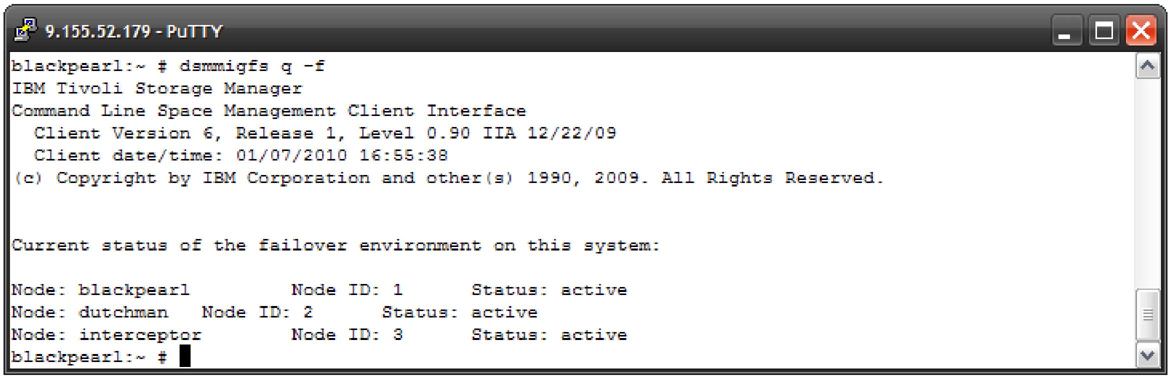
```
root@borodin:/gpfs> dsmc selective /gpfs/file -auditlogging=full -auditlogname=/tmp/audit.log >> /tmp/out
root@borodin:/gpfs> rm file
root@borodin:/gpfs> dsmc restore /gpfs/file -auditlogging=full -auditlogname=/tmp/audit.log >> /tmp/out
root@borodin:/gpfs> cat /tmp/audit.log
11/18/09   07:51:37 TSM Backup-Archive Client Version 6, Release 1, Level 3.0 091005C
11/18/09   07:51:39 ANS1650I Command: selective /gpfs/file
11/18/09   07:51:40 ANS1651I Backed Up: /gpfs/
11/18/09   07:51:40 ANS1651I Backed Up: /gpfs/file
11/18/09   07:51:57 TSM Backup-Archive Client Version 6, Release 1, Level 3.0 091005C
11/18/09   07:51:59 ANS1650I Command: restore /gpfs/file
11/18/09   07:51:59 ANS1655I Restored: /gpfs/file
root@borodin:/gpfs>
```

Test whether the expiration of files is working correctly on the TSM server (the restore of the backup copy shouldn't be possible after the file was expired on the server):

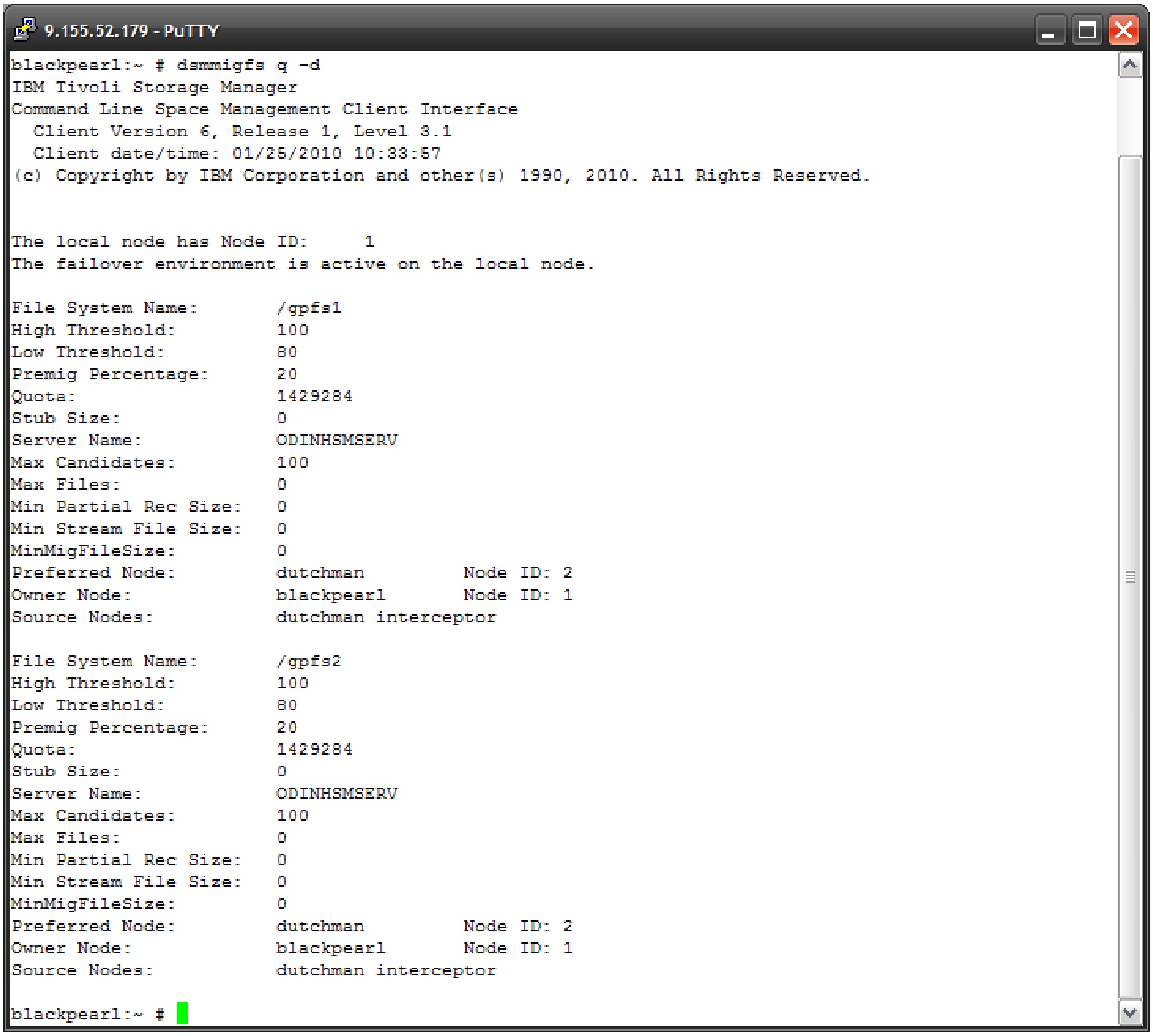


```
root@borodin:/gpfs> dsmc selective /gpfs/file -auditlogging=full -auditlogname=/tmp/audit.log >> /tmp/out
root@borodin:/gpfs> rm file
root@borodin:/gpfs> dsmc expire /gpfs/file -auditlogging=full -auditlogname=/tmp/audit.log >> /tmp/out
y
root@borodin:/gpfs> dsmc restore /gpfs/file -auditlogging=full -auditlogname=/tmp/audit.log >> /tmp/out
root@borodin:/gpfs> cat /tmp/audit.log
11/18/09   07:59:31 TSM Backup-Archive Client Version 6, Release 1, Level 3.0 091005C
11/18/09   07:59:31 ANS1650I Command: selective /gpfs/file
11/18/09   07:59:32 ANS1651I Backed Up: /gpfs/
11/18/09   07:59:32 ANS1651I Backed Up: /gpfs/file
11/18/09   07:59:53 TSM Backup-Archive Client Version 6, Release 1, Level 3.0 091005C
11/18/09   07:59:55 ANS1650I Command: expire /gpfs/file
11/18/09   07:59:58 ANS1657I Expired: /gpfs/file
11/18/09   08:00:10 TSM Backup-Archive Client Version 6, Release 1, Level 3.0 091005C
11/18/09   08:00:12 ANS1650I Command: restore /gpfs/file
root@borodin:/gpfs>
```

The HSM failover environment should be active on all nodes in the cluster. All cluster nodes that have HSM running on them should be reflected in the node section of the output:



```
blackpearl:~ # dsmmigfs q -f
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 6, Release 1, Level 0.90 IIA 12/22/09
  Client date/time: 01/07/2010 16:55:38
(c) Copyright by IBM Corporation and other(s) 1990, 2009. All Rights Reserved.


Current status of the failover environment on this system:

Node: blackpearl         Node ID: 1      Status: active
Node: dutchman   Node ID: 2      Status: active
Node: interceptor        Node ID: 3      Status: active
blackpearl:~ #
```

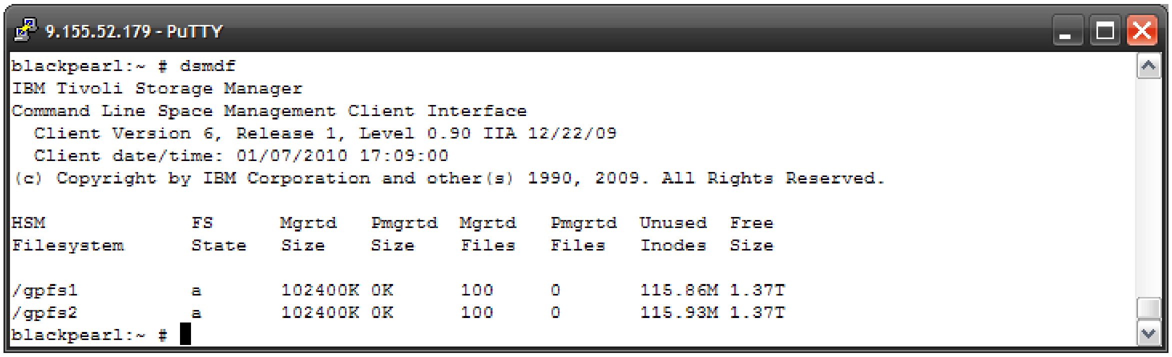


```
blackpearl:~ # dsmmigfs q -d
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 6, Release 1, Level 3.1
  Client date/time: 01/25/2010 10:33:57
(c) Copyright by IBM Corporation and other(s) 1990, 2010. All Rights Reserved.


The local node has Node ID:     1
The failover environment is active on the local node.

File System Name:       /gpfs1
High Threshold:         100
Low Threshold:          80
Premig Percentage:      20
Quota:                  1429284
Stub Size:              0
Server Name:            ODINHSMSERV
Max Candidates:         100
Max Files:              0
Min Partial Rec Size:   0
Min Stream File Size:   0
MinMigFileSize:         0
Preferred Node:         dutchman        Node ID: 2
Owner Node:             blackpearl      Node ID: 1
Source Nodes:           dutchman interceptor

File System Name:       /gpfs2
High Threshold:         100
Low Threshold:          80
Premig Percentage:      20
Quota:                  1429284
Stub Size:              0
Server Name:            ODINHSMSERV
Max Candidates:         100
Max Files:              0
Min Partial Rec Size:   0
Min Stream File Size:   0
MinMigFileSize:         0
Preferred Node:         dutchman        Node ID: 2
Owner Node:             blackpearl      Node ID: 1
Source Nodes:           dutchman interceptor

blackpearl:~ #
```

HSM management should be active for the file system. This is reflected in the FS State column.

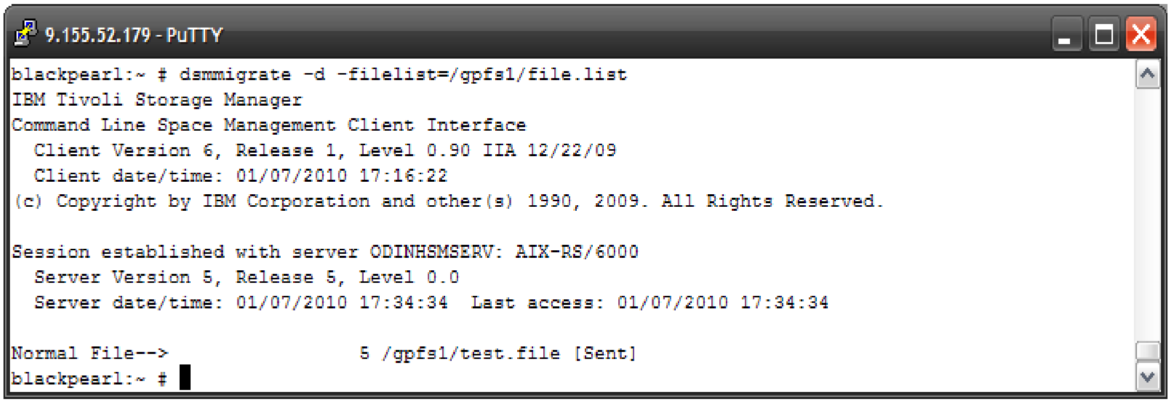
Note:

The command must be performed on the owner node of the file system.):



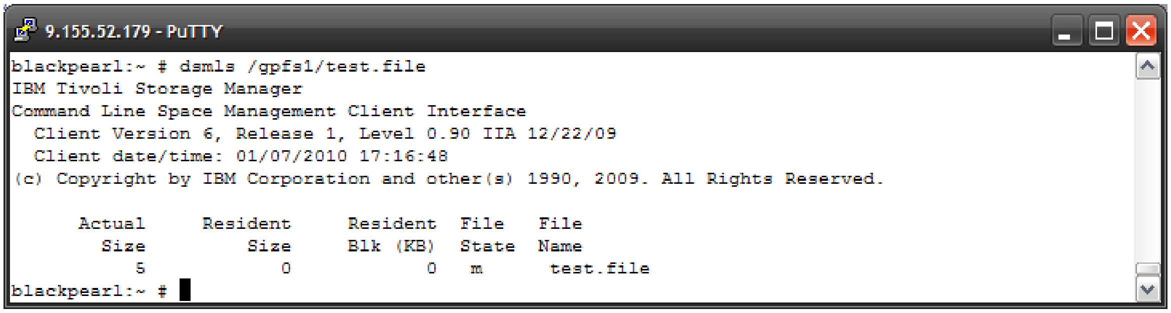
```
blackpearl:~ # dsmdf
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 6, Release 1, Level 0.90 IIA 12/22/09
  Client date/time: 01/07/2010 17:09:00
(c) Copyright by IBM Corporation and other(s) 1990, 2009. All Rights Reserved.

HSM             FS      Mgrtd   Pmgrtd  Mgrtd   Pmgrtd  Unused  Free
Filesystem      State   Size    Size    Files   Files   Inodes  Size

/gpfs1          a       102400K 0K      100     0       115.86M 1.37T
/gpfs2          a       102400K 0K      100     0       115.93M 1.37T
blackpearl:~ #
```

The base functionality, which will be invoked from the GPFS policy engine, should be ensured for the file system. That means it should be possible to migrate and recall files to the TSM server. Furthermore, the file list-based migration should work. This can be ensured with a simple test file migration. Just create a test file and put it on a file list. Then start the filelist migration by invoking the dsmmigrate command:



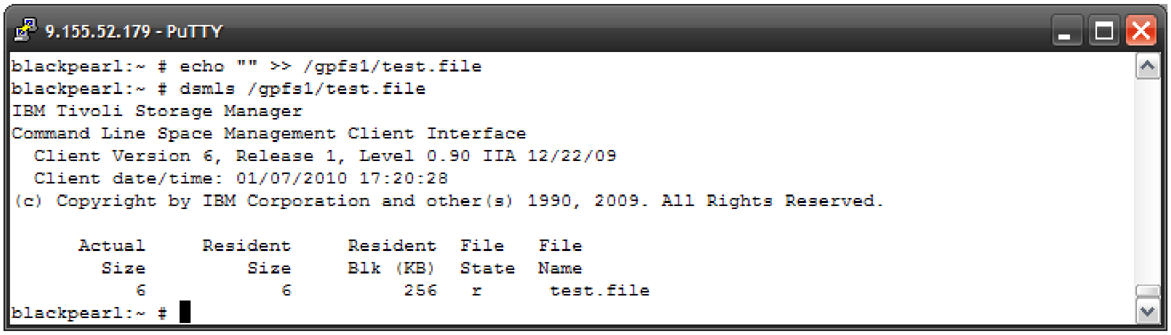
```
blackpearl:~ # dsmmigrate -d -filelist=/gpfs1/file.list
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 6, Release 1, Level 0.90 IIA 12/22/09
  Client date/time: 01/07/2010 17:16:22
(c) Copyright by IBM Corporation and other(s) 1990, 2009. All Rights Reserved.

Session established with server ODINHSMSERV: AIX-RS/6000
  Server Version 5, Release 5, Level 0.0
  Server date/time: 01/07/2010 17:34:34  Last access: 01/07/2010 17:34:34

Normal File-->                 5 /gpfs1/test.file [Sent]
blackpearl:~ #
```

The test file is now in file state migrated. You can use the dsmls command to verify the file state:



```
blackpearl:~ # dsmls /gpfs1/test.file
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 6, Release 1, Level 0.90 IIA 12/22/09
  Client date/time: 01/07/2010 17:16:48
(c) Copyright by IBM Corporation and other(s) 1990, 2009. All Rights Reserved.

      Actual     Resident     Resident  File   File
        Size         Size     Blk (KB)  State  Name
           5            0            0   m      test.file
blackpearl:~ #
```

If the file content changes, the TSM for Space Management client will recall the file and reset the file state to resident. You can use the following command sequence to verify this behavior:



```
blackpearl:~ # echo "" >> /gpfs1/test.file
blackpearl:~ # dsmls /gpfs1/test.file
IBM Tivoli Storage Manager
Command Line Space Management Client Interface
  Client Version 6, Release 1, Level 0.90 IIA 12/22/09
  Client date/time: 01/07/2010 17:20:28
(c) Copyright by IBM Corporation and other(s) 1990, 2009. All Rights Reserved.

      Actual     Resident     Resident  File   File
        Size         Size     Blk (KB)  State  Name
           6            6          256   r      test.file
blackpearl:~ #
```

Note:

Both clients, TSM for Space Management and the TSM Backup Archive, should use the same TSM server for the given file system. In this case, the TSM for Space Management functionality “inline copy” will work as designed. That means a file that was migrated before it is backed up will not need to be transferred again. The TSM Server will transfer the file internally from the space main pool to the backup pool. For more information, see the TSM for Space Management user guide.

Create proper policy rules for the threshold migration

The whole language definition of the GPFS policy engine can be found in the GPFS Advanced Administration Guide. To give an overview of the possible functionality, the TSM team has developed and tested the following rule set that can be used for policy-driven migration. The comment style is the same as the C++ comment style and it will be ignored by the GPFS policy engine. This is the recommended rule set:

```
define(
    is_premigrated,
    (MISC_ATTRIBUTES LIKE '%M%' AND MISC_ATTRIBUTES NOT LIKE '%V%')
)

define(
    is_migrated,
    (MISC_ATTRIBUTES LIKE '%V%')
)

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

define(
    mb_allocated,
    (INTEGER(KB_ALLOCATED / 1024))
)

/*=== a simple exclude list ===*/

define(
    exlude_list,
    (PATH_NAME LIKE '%/.SpaceMan/%'
    OR NAME LIKE '%dsmerror.log%'
    or NAME LIKE '.mmbackupShadow%')
)

define(
    weight_expression,
    (CASE

        /*=== The file is very young, the ranking is very low ===*/

        WHEN access_age <= 1  THEN 0

        /*=== The file is very small, the ranking is low ===*/

        WHEN mb_allocated < 1 THEN access_age

        /*=== The file is premigrated and large and old enough,
        the ranking is very high ===*/

        WHEN is_premigrated   THEN mb_allocated * access_age * 10

        /*=== The file is resident and large and old enough,
              the ranking is standard ===*/

        ELSE                       mb_allocated * access_age
    END)
)

RULE EXTERNAL LIST 'candidatesList' EXEC '/tmp/exec.list'
RULE EXTERNAL POOL 'hsm' EXEC '/tmp/exec.hsm'

/*=== create the candidates list and transfer it to the exec.list
      command (for logging) ===*/

RULE 'bestCandidates' LIST 'candidatesList'
    WEIGHT (weight_expression)
    SHOW (weight_expression)
    WHERE NOT (exlude_list)
        AND NOT (is_migrated)

/*=== create the candidates list and transfer it to the exec.hsm
      command for migration ===*/

RULE 'TM1' MIGRATE FROM POOL 'system'
        THRESHOLD(90,80,70)
        WEIGHT(weight_expression)
        TO POOL 'hsm'
        WHERE NOT (exlude_list)
            AND NOT (is_migrated)
```

It should be stored in a rule set file to make it usable for the policy engine commands. The following figure is a simple execute list shell script:

```
#!/usr/bin/ksh
if [ $# -ne 2 ]
   then echo "USAGE : $0 <[LIST|TEST]> <fileList>"
   exit 1
fi
case $1 in
    LIST)
        echo
        echo "INVOKED LIST COMMAND : `date +%c`"
        filename="/tmp/policy.out.`date +%s`"
        cat $2 >> $filename
        lines=`cat $filename | wc -l`
        echo "LINES WRITTEN        : $lines"
        echo "OUT FILE NAME        : $filename"
        echo
        rc=0
        ;;
    TEST)
```

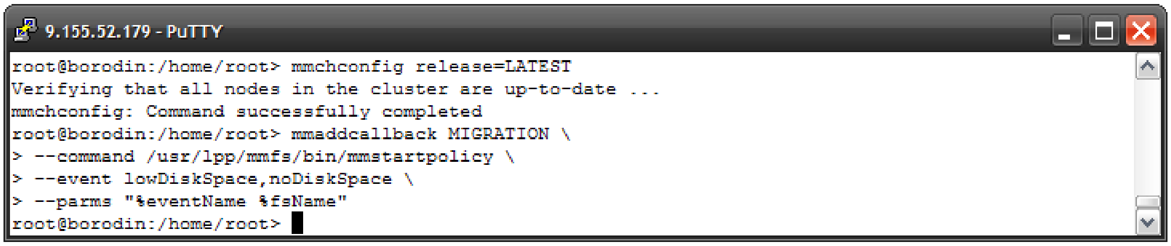
```
        echo "TEST LIST COMMAND     : `date +%c`"
        rc=0
        ;;
       *)
        echo "UNKNOWN INPUT         : `date +%c`"
        rc=1
        ;;
esac
exit $rc
```

Note:

The GPFS policy includes the logic for the GPFS-driven threshold migration. It is essential to define a policy that is dynamic enough to give GPFS the ability to migrate files until the low threshold is reached for the file system. It is recommended to not use fixed values (e.g. file size > 10MB). This can lead to policy scans being run endlessly.

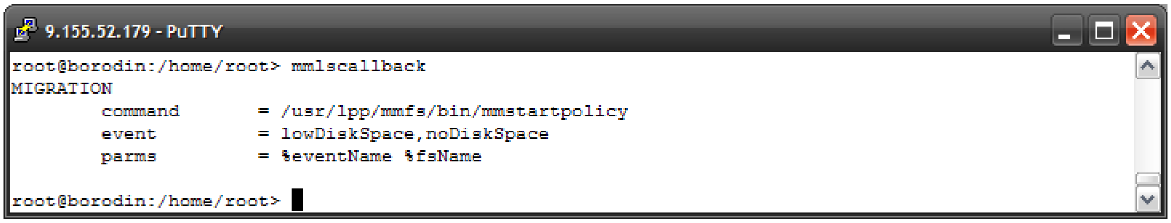
Configure the policy engine for automatic threshold migration

GPFS provides the callback functionality to register a user-defined command that GPFS will execute when certain events occur. This functionality can be used to register the policy engine to be run in case of a high threshold in a file system. The function must be enabled as shown below:



```
root@borodin:/home/root> mmchconfig release=LATEST
Verifying that all nodes in the cluster are up-to-date ...
mmchconfig: Command successfully completed
root@borodin:/home/root> mmaddcallback MIGRATION \
> --command /usr/lpp/mmfs/bin/mmstartpolicy \
> --event lowDiskSpace,noDiskSpace \
> --parms "%eventName %fsName"
root@borodin:/home/root>
```

To verify if the callback was set correctly, GPFS provides a special ls command. It should display the following callback settings:

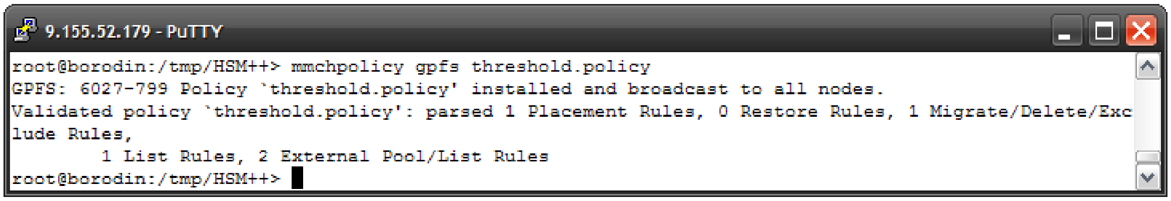


```
root@borodin:/home/root> mmlscallback
MIGRATION
        command       = /usr/lpp/mmfs/bin/mmstartpolicy
        event         = lowDiskSpace,noDiskSpace
        parms         = %eventName %fsName

root@borodin:/home/root>
```

The mmstartpolicy script invokes the mmapplypolicy command. It is possible to set a user-defined startscript instead of the predefined mmstartpolicy script.

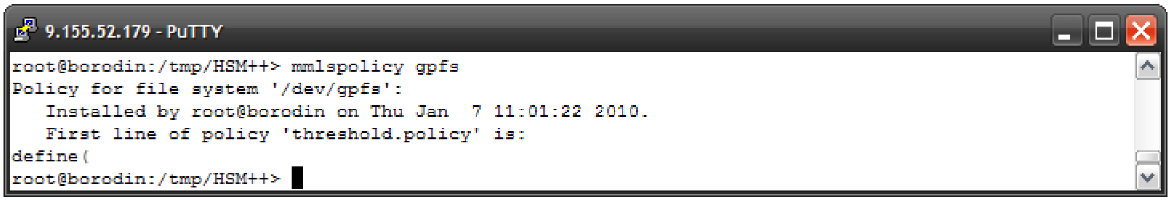
Use GPFS to monitor the file system thresholds

It is mandatory that the rule set, which should be used for threshold migration, includes a MIGRATE rule with a valid threshold setting. (*See the GPFS Advanced Administration Guide for more details).* Also the recommended rule above can be used for the threshold migration. To enable the rule set, the following command must be invoked:



```
root@borodin:/tmp/HSM++> mmchpolicy gpfs threshold.policy
GPFS: 6027-799 Policy `threshold.policy' installed and broadcast to all nodes.
Validated policy `threshold.policy': parsed 1 Placement Rules, 0 Restore Rules, 1 Migrate/Delete/Exc
lude Rules,
        1 List Rules, 2 External Pool/List Rules
root@borodin:/tmp/HSM++>
```

The policy settings can be verified by using the GPFS command mmlspolicy. (The command paramter -L will cause the output of the whole installed rule set content).



```
root@borodin:/tmp/HSM++> mmlspolicy gpfs
Policy for file system '/dev/gpfs':
   Installed by root@borodin on Thu Jan  7 11:01:22 2010.
   First line of policy 'threshold.policy' is:
define(
root@borodin:/tmp/HSM++>
```

Note:

It is possible to set the high threshold for the file system to a value below the current file system usage value. That will trigger an immediate scan and threshold migration until the rule set is installed for the specified device.

Define the number of migration processes started by the policy engine

Per default, the policy engine will start 24 migration processes on each node. The default number of files per file list is 100. It is possible to define these values by using the mmapplypolicy arguments -m for the number of processes and -B for the number of files per file list. By using the mmapplypolicy argument –N the node names can be specified which should be used to perform both the policy scan and the migration.

Note:

In the case of the default settings for the number of migration processes started by the GPFS policy engine, it is essential to modify TSM Server settings. The default value for allowed TSM server sessions is 25. If the policy engine runs in a single node environment there is just one possible session left for transparent recalls or backup runs. The parameter should be increased to a higher value that fits in your environment. *For more information see the TSM Server Admin Guide.*

Note:

Using the mmapplypolicy arguments described above will not change the default values for the threshold migration started by the automatic monitoring of the GPFS policy engine. In this case the predefined script mmstartpolicy must be modified.