General Parallel File System
Version 4 Release 1

Advanced Administration Guide



SC23-7032-00

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032

General Parallel File System
Version 4 Release 1

Advanced Administration Guide



SC23-7032-00

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 3

Note
Before using this information and the product it supports, read the information in "Notices" on page 283.

This edition applies to version 4 release 1 modification 0 of the following products, and to all subsequent releases and modifications until otherwise indicated in new editions:

- IBM General Parallel File System ordered through Passport Advantage® (product number 5725-Q01)
- IBM General Parallel File System ordered through AAS/eConfig (product number 5641-GPF)
- IBM General Parallel File System ordered through HVEC/Xcel (product number 5641-GP6, 5641-GP7, or 5641-GP8)

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

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

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

Contents

Figures vii

Tables ix

About this information xi
Prerequisite and related information xii
Conventions used in this information xiii
How to send your comments xiv

Summary of changes xv

Chapter 1. Accessing GPFS file systems from other GPFS clusters 1
User access to a GPFS file system owned by another GPFS cluster 3
Mounting a file system owned and served by another GPFS cluster 4
Managing remote access to GPFS file systems . . . 6
Using remote access with public and private IP addresses 6
Using multiple security levels 8
Changing security keys 9
NIST compliance 10
Additional information about GPFS file systems accessed by nodes that belong to other GPFS clusters 10

Chapter 2. Information Lifecycle Management for GPFS 13
Storage pools 13
Internal GPFS storage pools 14
External storage pools 18
Policies and rules 19
Policies 19
Policy rules 20
Semantics of the mmapplypolicy command and its policy rules 36
Policy rules - examples and tips 39
Managing policies 44
Working with external storage pools 45
Backup and restore with storage pools 50
Filesets 52
Fileset namespace 52
Filesets and quotas 53
Filesets and storage pools 53
Filesets and global snapshots 53
Fileset-level snapshots 54
Filesets and backup 54
Managing filesets 55
Immutability and appendOnly restrictions 58

Chapter 3. Creating and maintaining snapshots of GPFS file systems 61
Creating your GPFS snapshot 61
Listing GPFS snapshots 62
Restoring a GPFS file system from a snapshot . . . 63
Using the policy engine to read a snapshot 64
Linking to your GPFS snapshots 64
Deleting your GPFS snapshot 65

Chapter 4. Creating and managing file clones 67
Creating file clones 67
Listing file clones 68
Deleting file clones 69
Splitting file clones from clone parents 69
File clones and disk space management 69
File clones and snapshots 69
File clones and policy files 70

Chapter 5. Scale Out Backup and Restore (SOBAR) 71
Backup procedure with SOBAR 71
Restore procedure with SOBAR 72

Chapter 6. Establishing disaster recovery for your GPFS cluster 75
Synchronous mirroring utilizing GPFS replication . 76
Setting up a GPFS cluster with synchronous mirroring utilizing GPFS replication 77
Steps to take after a disaster when using GPFS replication 78
Synchronous mirroring utilizing IBM TotalStorage ESS PPRC 81
An active/active GPFS cluster 82
An active/passive GPFS cluster 86
Data integrity and the use of PPRC consistency groups 89
Asynchronous mirroring utilizing ESS FlashCopy . . 90
Setting up FlashCopy using file-system-level suspension 94

Chapter 7. Implementing a clustered NFS using GPFS on Linux 95
NFS monitoring 95
NFS failover 95
NFS locking and load balancing 95
CNFS network setup 96
CNFS setup 96
CNFS administration 97

Chapter 8. Performance and health monitoring 99
Network performance monitoring 99

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 5

Chapter 9. Monitoring GPFS I/O performance with the mmpmon command 103
Overview of mmpmon 103
Specifying input to the mmpmon command . . . 103
Running mmpmon on multiple nodes 104
Running mmpmon concurrently from multiple users on the same node 104
Display I/O statistics per mounted file system 104
Display I/O statistics for the entire node . . . 106
Reset statistics to zero 107
Understanding the node list facility 107
Add node names to a list of nodes for mmpmon processing 108
Delete a node list 109
Create a new node list 110
Show the contents of the current node list . . . 110
Delete node names from a list of nodes for mmpmon processing 111
Node list examples and error handling 112
Understanding the request histogram facility 115
Changing the request histogram facility request size and latency ranges 117
Disabling the request histogram facility. . . . 118
Enabling the request histogram facility 119
Displaying the request histogram facility pattern 120
Resetting the request histogram facility data to zero 122
Displaying the request histogram facility statistics values. 123
Displaying mmpmon version 125
Example mmpmon scenarios and how to analyze and interpret their results 126
fs_io_s and io_s output - how to aggregate and analyze the results. 126
Request histogram (rhist) output - how to aggregate and analyze the results. 128
Using request *source* and prefix directive *once* 129
Other information about mmpmon output. . . . 134
Counter sizes and counter wrapping 135
Return codes from mmpmon 135

Chapter 10. GPFS SNMP support . . . 137
Installing Net-SNMP 137
Configuring Net-SNMP 138
Configuring management applications 138
Installing MIB files on the collector node and management node. 139
Collector node administration 139
Starting and stopping the SNMP subagent. . . . 140
The management and monitoring subagent . . . 140
SNMP object IDs 141
MIB objects 141
Cluster status information 141
Cluster configuration information 142
Node status information. 142
Node configuration information 142
File system status information. 143
File system performance information 144
Storage pool information 145
Disk status information 145
Disk configuration information 146
Disk performance information. 146
Net-SNMP traps 147

Chapter 11. Identity management on Windows 149
Auto-generated ID mappings 149
Installing Windows IMU 149
Configuring ID mappings in IMU 150

Chapter 12. GPFS Native RAID (GNR) 153
Overview. 153
GPFS Native RAID features 154
RAID codes 154
End-to-end checksum 155
Declustered RAID 155
Disk configurations 157
Recovery groups 157
Declustered arrays. 158
Virtual and physical disks 159
Virtual disks. 159
Physical disks 159
Solid-state disks 160
Disk hospital 160
Health metrics 160
Pdisk discovery 160
Disk replacement 160
Managing GPFS Native RAID 161
Recovery groups 161
Pdisks 162
Declustered arrays. 165
Vdisks. 167
Maintenance. 169
Component configuration in the GPFS Storage Server 172
Overall management of GPFS Native RAID . . 178
GPFS Native RAID setup and disk replacement on the IBM Power 775 Disk Enclosure 182
Example scenario: Configuring GPFS Native RAID recovery groups 182
Example scenario: Replacing failed disks in a Power 775 Disk Enclosure recovery group . . . 194
GPFS Native RAID setup and maintenance on the IBM System x GPFS Storage Server (GSS) 200
Updating firmware on enclosures and drives 200
Example scenario: Configuring GPFS Native RAID recovery groups on the GSS 201
Example scenario: Replacing failed disks in a GSS recovery group 218
Example scenario: Replacing failed GSS storage enclosure components 223
Example scenario: Replacing a failed GSS storage enclosure drawer 224
Example scenario: Replacing a failed GSS storage enclosure 226
Example scenario: Checking the health of a GSS configuration 226

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 6

Chapter 13. Active file management 229
| Active file management architecture 229
Caching modes 230
File system caching and synchronization 232
Asynchronous operations 232
Synchronous operations 232
Update synchronization 232
Components of a cluster that is running AFM . . 233
Global namespace 234
Cache eviction 234
Disconnected operation 235
Expiration 235
| Cache states 235
Failure and recovery 237
Steps to deal with an IW cache fileset disaster . . 238
Prefetching 239
Peer snapshots 239
Viewing snapshots at home 240
Failover of cache filesets 240
Partial file caching 240
| Parallel I/O 241
Disabling AFM 242
Hierarchical storage management and AFM . . . 242
mmbackup and AFM 242
| AFM-based NFS migration 242
Restrictions in active file management 245
Home cluster errors 246
Administrative actions 247
System setup and requirements 247
Dealing with requeued messages 249
Tuning active file management home communications 249

Chapter 14. GPFS File Placement Optimizer 255
Distributing data across a cluster 258
FPO pool file placement and AFM 258
Restrictions 258

| Chapter 15. Encryption 261
| Encryption keys 261
| Encryption policies 262
| Encryption policy rules 262
| Encryption setup requirements 267
| Establishing an encryption-enabled environment 269
| Secure deletion 272
| Encryption and FIPS compliance 274
| Encryption and NIST compliance 274
| Encryption and backup/restore 274
| Encryption and snapshots 274

Chapter 16. Miscellaneous advanced administration topics 275
Changing IP addresses and host names 275
Enabling a cluster for IPv6 276
Using multiple token servers 276
Exporting file system definitions between clusters 277
GPFS port usage 277

Accessibility features for GPFS . . . 281
Accessibility features 281
Keyboard navigation 281
IBM and accessibility 281

Notices 283
Trademarks 284

Glossary 287

Index 293

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 7

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 8

Figures

1. Remote mount of a file system using NSD server access 1
2. Remote mount of a file system using SAN-attached disks 2
3. Multi-cluster configuration with multiple NSD servers 2
4. Use of public and private IP addresses in three GPFS clusters 8
5. Synchronous mirroring utilizing GPFS replication 77
6. A synchronous active/active PPRC-based mirrored GPFS configuration with a tiebreaker site 83
7. A synchronous active/passive PPRC-based GPFS configuration without a tiebreaker site . 86
8. Violation of write ordering without the use of a PPRC consistency group 90
9. High-level organization of a FlashCopy/PPRC recovery environment 93
10. Node running mmpmon 103
11. Properties window. 150
12. Redundancy codes supported by GPFS Native RAID. 154
13. Conventional RAID versus declustered RAID layouts. 156
14. Lower rebuild overhead in conventional RAID versus declustered RAID 157
15. GPFS Native RAID server and recovery groups in a ring configuration 158
16. Minimal configuration of two GPFS Native RAID servers and one storage JBOD 158
17. Example of declustered arrays and recovery groups in storage JBOD 159
18. Global namespace implemented using AFM 234

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 9

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 10

Tables

1. GPFS library information units xi
2. Conventions xiii
3. Summary of commands to set up cross-cluster file system access. 6
4. The effects of file operations on an immutable file or an appendOnly file. 59
5. Input requests to the **mmpmon** command 104
6. Keywords and values for the **mmpmon fs_io_s** response 105
7. Keywords and values for the **mmpmon io_s** response 106
8. Keywords and values for the **mmpmon reset** response 107
9. nlist requests for the **mmpmon** command 108
10. Keywords and values for the **mmpmon nlist add** response. 108
11. Keywords and values for the **mmpmon nlist del** response 109
12. Keywords and values for the **mmpmon nlist new** response 110
13. Keywords and values for the **mmpmon nlist s** response. 110
14. Keywords and values for the **mmpmon nlist** failures. 114
15. Keywords and values for the **mmpmon rhist nr** response 117
16. Keywords and values for the **mmpmon rhist off** response 119
17. Keywords and values for the **mmpmon rhist on** response 119
18. Keywords and values for the **mmpmon rhist p** response 120
19. Keywords and values for the **mmpmon rhist reset** response 123
20. Keywords and values for the **mmpmon rhist s** response 123
21. Keywords and values for the **mmpmon ver** response 126
22. gpfsClusterStatusTable: Cluster status information 141
23. gpfsClusterConfigTable: Cluster configuration information 142
24. gpfsNodeStatusTable: Node status information 142
25. gpfsNodeConfigTable: Node configuration information 142
26. gpfsFileSystemStatusTable: File system status information 143
27. gpfsFileSystemPerfTable: File system performance information. 144
28. gpfsStgPoolTable: Storage pool information 145
29. gpfsDiskStatusTable: Disk status information 145
30. gpfsDiskConfigTable: Disk configuration information 146
31. gpfsDiskPerfTable: Disk performance information 146
32. Net-SNMP traps 147
33. Pdisk states 164
34. Vdisk states 169
35. Background tasks 170
36. Keywords and descriptions of values provided in the **mmpmon vio_s** response . . 180
37. GPFS Native RAID callbacks and parameters 181
38. NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and non-default operating system I/O size for permitted GPFS Native RAID vdisks 184
39. AFM modes 230
40. Active file management configuration parameters and default values 251
41. Valid *EncParamString* values 263
42. Valid combine parameter string values 263
43. Valid wrapping parameter string values 263
44. GPFS port usage 278

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 11

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 12

About this information

This edition applies to GPFS™ version 4.1 for AIX®, Linux, and Windows.

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

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

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

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

To find out which version of GPFS is running on a particular Windows node, open the **Programs and Features** control panel. The IBM® General Parallel File System installed program name includes the version number.

Which GPFS information unit provides the information you need?

The GPFS library consists of the information units listed in Table 1.

To use these information units effectively, you must be familiar with the GPFS licensed product 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.

Table 1. GPFS library information units

Information unit	Type of information	Intended users
GPFS: 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
GPFS: Advanced Administration	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

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 13

Table 1. GPFS library information units (continued)

Information unit	Type of information	Intended users
GPFS: Concepts, Planning, and Installation	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
GPFS: 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
GPFS: 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 *GPFS: Concepts, Planning, and Installation*

Prerequisite and related information

For updates to this information, see the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfsbooks.html) or GPFS in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/welcome).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 14

For the latest support information, see the GPFS FAQ in the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) or GPFS FAQ in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfsclustersfaq.html).

Conventions used in this information

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

Note: Users of GPFS 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 name space starts under the `%SystemDrive%\cygwin` directory, so the GPFS cluster configuration data is stored in the `C:\cygwin\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.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 15

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 GPFS 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 GPFS development organization, send your comments to the following e-mail address:

gpfs@us.ibm.com

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 16

Summary of changes

This topic summarizes changes to the GPFS licensed program and the GPFS 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.

GPFS version 4 release 1

Changes to the GPFS licensed program and the GPFS library for version 4, release 1 include the following:

GPFS product structure
GPFS now comes in three levels of function: GPFS Standard Edition, GPFS Express Edition, and GPFS Advanced Edition.

Active file management (AFM)
Enhancements to AFM include the following:

- AFM environments can now support Parallel I/O. During reads, all mapped gateway nodes are used to fetch a single file from home. During writes, all mapped gateways are used to synchronize file changes to home.
- In addition to the NFS protocol, AFM now supports the native GPFS protocol for the AFM communication channel providing improved integration of GPFS features and attributes.
- GPFS 4.1 includes a number of features optimizing AFM operations and usability. These features include prefetch enhancements to handle gateway node failures during prefetch. AFM introduces new version of hashing (**afmHashVersion=2**), which minimizes the impact of gateway nodes joining or leaving the active cluster. Also, AFM cache states will now have different states based on fileset and queue states.
- GPFS 4.1 supports the migration of data from any legacy NFS storage device or GPFS cluster to an AFM fileset. Data migration eases data transfer when upgrading hardware or buying a new system. The data source is an NFS v3 export and can be either a GPFS or a non-GPFS source as well. AFM based migration can minimize downtime for applications and consolidate data from multiple legacy systems into a more powerful cache.

Autonomic tuning for mmbackup
The **mmbackup** command can be tuned to control the numbers of threads used on each node to scan the file system, perform inactive object expiration, and carry out modified object backup. In addition, the sizes of lists of objects expired or backed up can be controlled or autonomically tuned to select these list sizes if they are not specified. List sizes are now independent for backup and expire tasks. For more information, see the *GPFS: Administration and Programming Reference* topic: "Tuning backups with the mmbackup command".

Backup 3.2 format discontinued
Starting with GPFS 4.1, the **mmbackup** command will no longer support incremental backup using the */Device*/**.snapshots**/**.mmbuSnapshot** path name that was used with GPFS 3.2 and earlier. For more information, see the *GPFS: Administration and Programming Reference* topic: "File systems backed up using GPFS 3.2 or earlier versions of mmbackup".

Cluster Configuration Repository (CCR)
GPFS 4.1 introduces a new quorum-based repository for the configuration data. This replaces the current server-based repository, which required specific nodes to be designated as primary and backup configuration server nodes.

Cluster NFS improvements
Cluster NFS (CNFS) has been enhanced to support IPv6 and NFS V4.

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 17

Cygwin replaces SUA for Windows nodes
SUA is no longer supported for Windows nodes. Cygwin is now a required prerequisite before installing GPFS on Windows nodes.

Deadlock amelioration
Automated deadlock detection, automated deadlock data collection, and automated deadlock breakup can be used to help simplify deadlock troubleshooting.

Encryption
Support is provided for file encryption that ensures both secure storage and secure deletion of data. Encryption is only available with the GPFS Advanced Edition. For more information, see the *GPFS: Advanced Administration Guide* topic: "Encryption".

File Placement Optimizer (FPO)
Enhancements to FPO include the following:

- To avoid performance impacts, data locality is now maintained when running the **mmrestripefs -r** command.
- Asynchronous I/O performance was improved.
- The performance of GPFS **O_DIRECT** vectored I/O was improved.
- Data locality performance for AFM-FPO was improved.
- The **mmchpool** command was provided to change GPFS-FPO relevant attributes (**writeAffinityDepth** and **blockGroupFactor**) for FPO storage pools.
- Write affinity depth of 2 was improved to assign (write) all of the files in a fileset to the same second-replica node.

Fileset snapshot restore
Files can be restored from a fileset-level snapshot in a mounted file system.

Local read-only cache
Support is provided for large local read-only cache using solid-state disks. This makes data available with very low latency, and the cache serves to reduce the load on the shared network and on the backend disk storage, optimizing performance.

Message logging
Starting with GPFS 4.1, many GPFS log messages can be sent to **syslog** on Linux. Severity tags were added to numerous messages, and these tags can be used to filter the messages that are sent to **syslog**. The **systemLogLevel** attribute of the **mmchconfig** command controls which GPFS log messages are sent to **syslog**.

mmsetquota command
The new **mmsetquota** command enables you to set quota limits, default quota limits, or grace periods for users, groups, and filesets in the file system from which the command is issued.

NFSv4 ACL formats
The ACL entry **MKDIR** was replaced by **APPEND/MKDIR** to allow **WRITE** and **APPEND** to be specified independently. A new **NoPropagateInherit** ACL flag was introduced; this flag indicates that the ACL entry should be included in the initial ACL for subdirectories created in this directory but not further propagated to subdirectories created below that level.

NIST SP800-131A compliance
GPFS can be configured to operate in conformance with the NIST SP800-131A recommendations for communication across nodes when the **cipherList** configuration variable is specified. The **nistCompliance** configuration variable controls whether conformance with NIST SP800-131A is enforced. The algorithms and key lengths used for file encryption all conform with NIST SP800-131A.

NSD formats
A new NSD format was introduced. The new format is referred to as NSD v2, and the old format

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 18

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.

Online migration of extended attributes
The **mmmigratefs** command can now be run with the file system mounted.

Quota management
Quota management improvements for file system format 4.1 and higher include:

- Allowing quota management to be enabled and disabled without unmounting the file system.
- The **user.quota**, **group.quota**, and **fileset.quota** files are no longer used. Quota files are now metadata files and do not appear in the file system name space.

Rapid repair
Performance of repairing large replicated files when restarting disks which were down has been improved. The repair will occur only on the blocks that changed while the disk was down, rather than on the entire file.

snapshotCreated callback
A new event called **snapshotCreated** was added to help correlate the timing of DMAPI events with the creation of a snapshot.

TSM version verification
The TSM Backup-Archive client must be installed and at the same version on all the nodes that will execute the **mmbackup** command or named in a node specification with **-N**. Starting with GPFS 4.1, the **mmbackup** command will verify that the TSM Backup-Archive client versions and configuration are correct before executing the backup.

User-defined node classes
Nodes can now be grouped into user-defined node classes that are created with the **mmcrnodeclass** command. After a node class is created, it can be specified as an argument on commands that accept the **-N** *NodeClass* option. User-defined node classes are managed with the **mmchnodeclass**, **mmdelnodeclass**, and **mmlsnodeclass** commands.

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

New commands:
The following commands are new:

- **mmafmconfig**
- **mmchnodeclass**
- **mmchpool**
- **mmcrnodeclass**
- **mmdelnodeclass**
- **mmlsnodeclass**
- **mmsetquota**

New structures:
There are no new structures.

New subroutines:
There are no new subroutines.

Changed commands:
The following commands were changed:

- **mmaddcallback**
- **mmafmctl**
- **mmafmlocal**
- **mmauth**

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 19

- **mmbackup**
- **mmchattr**
- **mmchcluster**
- **mmchconfig**
- **mmchfileset**
- **mmchfs**
- **mmchpdisk**
- **mmchrecoverygroup**
- **mmcrcluster**
- **mmcrfileset**
- **mmcrfs**
- **mmcrnsd**
- **mmcrrecoverygroup**
- **mmcrvdisk**
- **mmdelvdisk**
- **mmdiag**
- **mmlscluster**
- **mmlsfs**
- **mmlsmount**
- **mmlsrecoverygroup**
- **mmmigratefs**
- **mmmount**
- **mmrestorefs**
- **mmsnapdir**
- **mmumount**

Changed structures:

The following structures were changed:

- **gpfs_acl_t**
- **gpfs_direntx_t**
- **gpfs_direntx64_t**
- **gpfs_iattr_t**
- **gpfs_iattr64_t**

Changed subroutines:

The following subroutines were changed:

- **gpfs_fgetattrs()**
- **gpfs_fputattrs()**
- **gpfs_fputattrswithpathname()**
- **gpfs_fstat()**
- **gpfs_stat()**

Deleted commands:

The following commands were deleted:

- **mmafmhomeconfig**

Deleted structures:

There are no deleted structures.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 20

Deleted subroutines:

There are no deleted subroutines.

Messages

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

New messages

6027-680, 6027-760, 6027-873, 6027-939, 6027-953, 6027-954, 6027-955, 6027-956, 6027-957, 6027-959, 6027-960, 6027-1629, 6027-1743, 6027-1744, 6027-1745, 6027-1746, 6027-1747, 6027-1748, 6027-1749, 6027-1750, 6027-1751, 6027-1752, 6027-2229, 6027-2230, 6027-2231, 6027-2232, 6027-2233, 6027-2728, 6027-2767, 6027-2793, 6027-2794, 6027-2795, 6027-2796, 6027-2797, 6027-2798, 6027-2822, 6027-2823, 6027-2824, 6027-2825, 6027-2826, 6027-2827, 6027-2828, 6027-2956, 6027-2957, 6027-2958, 6027-3066, 6027-3067, 6027-3068, 6027-3069, 6027-3070, 6027-3071, 6027-3072, 6027-3073, 6027-3074, 6027-3075, 6027-3076, 6027-3077, 6027-3078, 6027-3079, 6027-3080, 6027-3241, 6027-3242, 6027-3243, 6027-3244, 6027-3245, 6027-3246, 6027-3247, 6027-3248, 6027-3249, 6027-3250, 6027-3252, 6027-3253, 6027-3254, 6027-3304, 6027-3305, 6027-3404, 6027-3450, 6027-3451, 6027-3452, 6027-3453, 6027-3457, 6027-3458, 6027-3459, 6027-3460, 6027-3461, 6027-3462, 6027-3463, 6027-3464, 6027-3465, 6027-3466, 6027-3468, 6027-3469, 6027-3470, 6027-3471, 6027-3472, 6027-3473, 6027-3474, 6027-3475, 6027-3476, 6027-3477, 6027-3478, 6027-3479, 6027-3480, 6027-3481, 6027-3482, 6027-3483, 6027-3484, 6027-3485, 6027-3486, 6027-3487, 6027-3488, 6027-3489, 6027-3490, 6027-3491, 6027-3493, 6027-3494, 6027-3495, 6027-3496, 6027-3497, 6027-3498, 6027-3499, 6027-3500, 6027-3501, 6027-3502, 6027-3503, 6027-3504, 6027-3505, 6027-3506, 6027-3509, 6027-3510, 6027-3511, 6027-3512, 6027-3513, 6027-3514, 6027-3515, 6027-3516, 6027-3517, 6027-3518, 6027-3519, 6027-3520, 6027-3521, 6027-3522, 6027-3524, 6027-3527, 6027-3528, 6027-3529, 6027-3530, 6027-3533, 6027-3534, 6027-3535, 6027-3536, 6027-3537, 6027-3540, 6027-3541, 6027-3543, 6027-3544, 6027-3545, 6027-3546, 6027-3547, 6027-3548, 6027-3549, 6027-3550, 6027-3555, 6027-3700, 6027-3701, 6027-3702, 6027-3703, 6027-3704

Changed messages

6027-328, 6027-542, 6027-573, 6027-575, 6027-595, 6027-597, 6027-755, 6027-882, 6027-884, 6027-885, 6027-886, 6027-906, 6027-907, 6027-910, 6027-996, 6027-1227, 6027-1260, 6027-1261, 6027-1262, 6027-1263, 6027-1292, 6027-1303, 6027-1305, 6027-1308, 6027-1309, 6027-1524, 6027-1717, 6027-1718, 6027-1890, 6027-1891, 6027-1898, 6027-2150, 6027-2158, 6027-2204, 6027-2205, 6027-2206, 6027-2741, 6027-2756, 6027-2758, 6027-3026, 6027-3060, 6027-3215, 6027-3226, 6027-3227, 6027-3228, 6027-3232, 6027-3236, 6027-3239, 6027-3240

Changed messages (only severity tags added)

6027-300, 6027-302, 6027-303, 6027-304, 6027-305, 6027-306, 6027-310, 6027-311, 6027-312, 6027-313, 6027-314, 6027-315, 6027-316, 6027-317, 6027-318, 6027-323, 6027-334, 6027-335, 6027-336, 6027-337, 6027-338, 6027-339, 6027-341, 6027-342, 6027-343, 6027-344, 6027-346, 6027-347, 6027-348, 6027-349, 6027-350, 6027-361, 6027-365, 6027-378, 6027-435, 6027-472, 6027-473, 6027-474, 6027-479, 6027-481, 6027-482, 6027-483, 6027-490, 6027-499, 6027-532, 6027-533, 6027-550, 6027-590, 6027-593, 6027-596, 6027-598, 6027-599, 6027-604, 6027-605, 6027-606, 6027-608, 6027-611, 6027-613, 6027-616, 6027-617, 6027-618, 6027-622, 6027-629, 6027-630, 6027-635, 6027-636, 6027-637, 6027-638, 6027-639, 6027-640, 6027-641, 6027-642, 6027-643, 6027-646, 6027-647, 6027-650, 6027-695, 6027-696, 6027-697, 6027-698, 6027-699, 6027-700, 6027-701, 6027-702, 6027-703, 6027-711, 6027-712, 6027-716, 6027-717, 6027-719, 6027-720, 6027-721, 6027-724, 6027-726, 6027-734, 6027-747, 6027-750, 6027-751, 6027-752, 6027-753, 6027-756, 6027-761, 6027-765, 6027-766, 6027-767, 6027-777, 6027-778, 6027-784, 6027-785, 6027-786, 6027-787, 6027-788, 6027-866, 6027-870, 6027-871, 6027-872, 6027-874, 6027-875, 6027-876, 6027-877, 6027-878, 6027-879, 6027-881, 6027-883, 6027-887, 6027-888, 6027-889, 6027-890, 6027-891, 6027-892, 6027-893, 6027-894, 6027-895, 6027-896, 6027-897, 6027-898, 6027-899, 6027-900, 6027-901, 6027-902, 6027-903, 6027-904, 6027-905, 6027-908, 6027-909, 6027-911, 6027-912, 6027-920, 6027-921, 6027-922, 6027-923, 6027-924, 6027-928, 6027-929, 6027-930, 6027-931, 6027-932, 6027-933, 6027-934, 6027-935, 6027-936, 6027-937, 6027-938, 6027-948, 6027-949, 6027-950, 6027-951, 6027-997, 6027-998, 6027-999, 6027-1500, 6027-1501, 6027-1502, 6027-1510, 6027-1511, 6027-1512, 6027-1537, 6027-1538, 6027-1539,

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 21

6027-1540, 6027-1541, 6027-1542, 6027-1544, 6027-1545, 6027-1546, 6027-1547, 6027-1548, 6027-1549, 6027-1550, 6027-1666, 6027-1709, 6027-1710, 6027-1711, 6027-1716, 6027-1724, 6027-1725, 6027-1726, 6027-1727, 6027-1728, 6027-1729, 6027-1730, 6027-1731, 6027-1732, 6027-1734, 6027-1735, 6027-1736, 6027-1737, 6027-1738, 6027-1739, 6027-1740, 6027-1741, 6027-1742, 6027-1803, 6027-1804, 6027-1805, 6027-1806, 6027-1807, 6027-1808, 6027-1809, 6027-1810, 6027-1811, 6027-1812, 6027-1813, 6027-1814, 6027-1815, 6027-1816, 6027-1817, 6027-1818, 6027-1824, 6027-1825, 6027-1851, 6027-1852, 6027-2049, 6027-2050, 6027-2576, 6027-2618, 6027-2621, 6027-2623, 6027-2667, 6027-2673, 6027-2674, 6027-2682, 6027-2694, 6027-2695, 6027-2696, 6027-2700, 6027-2706, 6027-2707, 6027-2708, 6027-2710, 6027-2711, 6027-2716, 6027-2722, 6027-2723, 6027-2724, 6027-2725, 6027-2726, 6027-2730, 6027-2734, 6027-2735, 6027-2740, 6027-2742, 6027-2744, 6027-2745, 6027-2746, 6027-2747, 6027-2750, 6027-2751, 6027-2752, 6027-2753, 6027-2754, 6027-2755, 6027-2757, 6027-2759, 6027-2760, 6027-2766, 6027-2777, 6027-2778, 6027-2779, 6027-2780, 6027-2781, 6027-2782, 6027-2783, 6027-2784, 6027-2785, 6027-2786, 6027-2787, 6027-2788, 6027-2789, 6027-2805, 6027-2806, 6027-2807, 6027-2810, 6027-2950, 6027-2952, 6027-2953, 6027-2954, 6027-2955, 6027-3035, 6027-3045, 6027-3058, 6027-3214, 6027-3224, 6027-3225, 6027-3229, 6027-3230, 6027-3233, 6027-3234, 6027-3235, 6027-3402

Deleted messages

6027-1264, 6027-1265, 6027-1311, 6027-1312, 6027-1323, 6027-1324, 6027-1325, 6027-1327, 6027-1328, 6027-1330, 6027-1336, 6027-1337, 6027-1354, 6027-1355, 6027-1356, 6027-1369, 6027-1376, 6027-1397, 6027-1558, 6027-1569, 6027-1580, 6027-1585, 6027-1586, 6027-1629, 6027-1698, 6027-1868, 6027-1920, 6027-1944, 6027-1965, 6027-1971, 6027-1972, 6027-1973, 6027-1991, 6027-3218, 6027-3219, 6027-3237, 6027-3238

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 22

Chapter 1. Accessing GPFS file systems from other GPFS clusters

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

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

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

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

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

Each node in the GPFS cluster requiring file system access must have either:

- A virtual connection to the file system data through an NSD server (refer to Figure 1).
- A physical connection to the disks containing file system data (refer to Figure 2 on page 2).

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

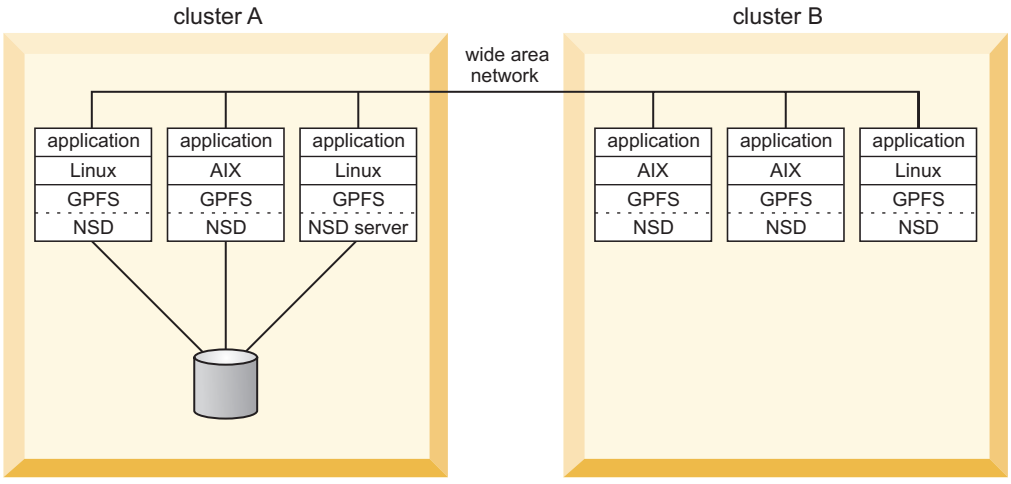


Figure 1. Remote mount of a file system using NSD server access

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 23

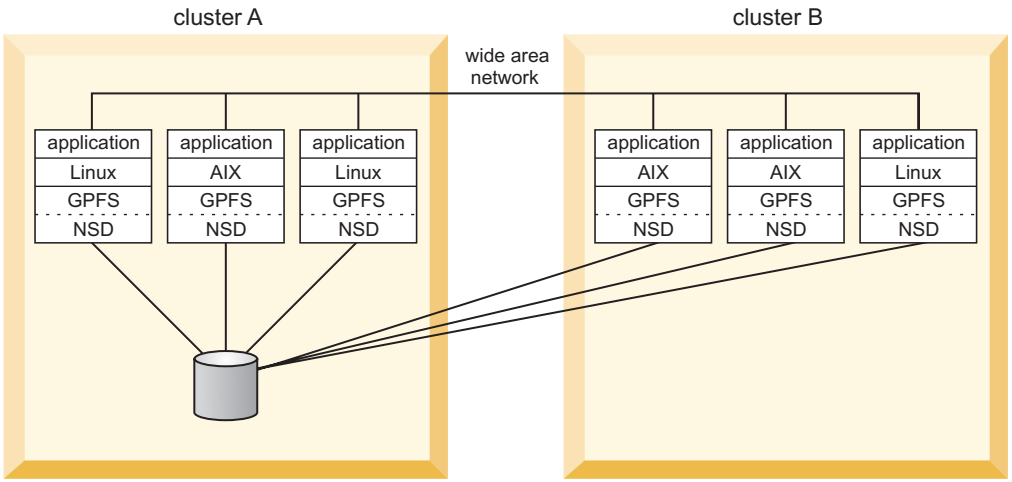


Figure 2. Remote mount of a file system using SAN-attached disks

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

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

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

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

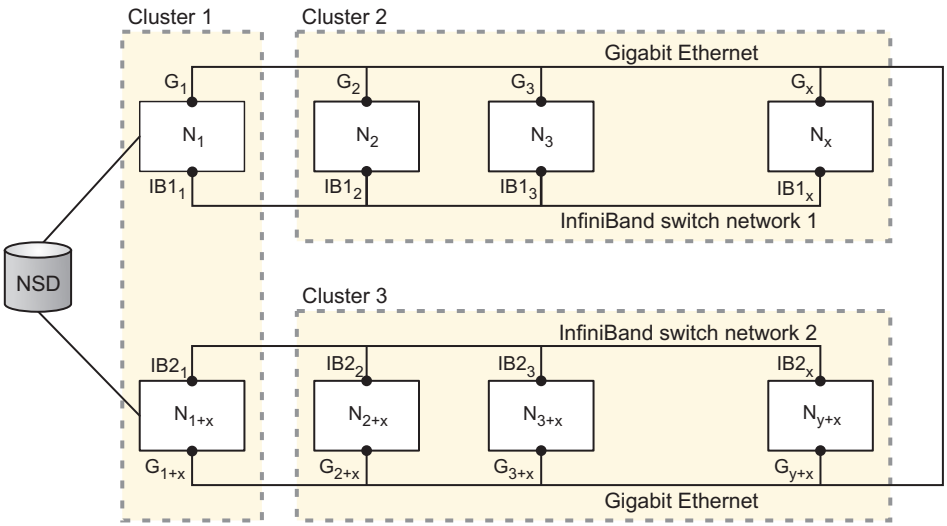


Figure 3. Multi-cluster configuration with multiple NSD servers

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 24

- "User access to a GPFS file system owned by another GPFS cluster"
- "Mounting a file system owned and served by another GPFS cluster" on page 4
- "Managing remote access to GPFS file systems" on page 6
- "Using remote access with public and private IP addresses" on page 6
- "Using multiple security levels" on page 8
- "Changing security keys" on page 9
- "Additional information about GPFS file systems accessed by nodes that belong to other GPFS clusters" on page 10

User access to a GPFS file system owned by another GPFS cluster

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

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

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

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 25

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

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

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

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

Mounting a file system owned and served by another GPFS cluster

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

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

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

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

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

   ```
   mmauth genkey new
   ```

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

   ```
   mmauth update . -l AUTHONLY
   ```

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

   ```
   mmauth genkey new
   ```

5. On **cluster2**, the system administrator enables authorization by issuing:

   ```
   mmauth update . -l AUTHONLY
   ```

6. The system administrator of **cluster2** gives file **/var/mmfs/ssl/id_rsa.pub** to the system administrator of **cluster1**. This operation requires the two administrators to coordinate their activities, and must occur outside of the GPFS command environment.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 26

7. On **cluster1**, the system administrator issues the **mmauth add** command to authorize **cluster2** to mount file systems owned by **cluster1** utilizing the key file received from the administrator of **cluster2**:

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

   where:

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

   cluster2_id_rsa.pub
   Is the name of the file obtained from the administrator of **cluster2** in Step 6 on page 4.
8. On **cluster1**, the system administrator issues the **mmauth grant** command to authorize **cluster2** to mount specific file systems owned by **cluster1**:

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

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

   where:

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

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

   cluster1_id_rsa.pub
   Is the name of the file obtained from the administrator of **cluster1** in Step 3 on page 4.

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

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

    where:

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

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

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

    /mygpfs
    Is the local mount point in **cluster2**.
11. On **cluster2**, the command:

    ```
    mmmount /dev/mygpfs
    ```

    then mounts the file system.

Table 3 on page 6 summarizes the commands that the administrators of the two clusters need to issue so that the nodes in **cluster2** can mount the remote file system **fs1**, owned by **cluster1**, assigning **rfs1** as the local name with a mount point of **/rfs1**.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 27

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

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

Managing remote access to GPFS file systems

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

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

```
mmauth show
```

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

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

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

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

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

```
mmauth delete cluster3
```

Using remote access with public and private IP addresses

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

Private IP addresses are on one of these subnets:

- 10.0.0.0
- 172.16.0.0
- 192.168.0.0

See *The Internet Engineering Task Force* and search on *RFC 1597 - Address Allocation for Private Internets*.

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

Figure 4 on page 8 describes an AIX cluster named **CL1** with nodes named **CL1N1**, **CL1N2**, and so forth, a Linux cluster named **CL2** with nodes named **CL2N1**, **CL2N2**, and another Linux cluster named **CL3** with a node named **CL3N1**. Both Linux clusters have public Ethernet connectivity, and a Gigabit Ethernet configured with private IP addresses (10.200.0.1 through 10.200.0.24), not connected to the public Ethernet. The Infiniband Switch on the AIX cluster **CL1** is configured using public IP addresses on the 7.2.24/13 subnet and is accessible from the outside.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 28

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

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

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

   GPFS assumes subnet specifications for private networks are independent between clusters (private networks are assumed not physically connected between clusters). The remaining steps show how to indicate that a private network is shared between clusters.
3. Using the **mmchconfig** command for the **CL2** cluster, add the **subnets='10.200.0.0/CL2.kgn.ibm.com 10.200.0.0/CL3.kgn.ibm.com'** attribute.

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

   The **10.200.0.0/CL2.kgn.ibm.com** portion allows all **CL2** nodes to communicate over their Gigabit Ethernet. The **10.200.0.0/CL3.kgn.ibm.com** portion allows remote mounts between clusters **CL2** and **CL3** to communicate over their Gigabit Ethernet.
4. Using the **mmchconfig** command for the **CL3** cluster, add the **subnets='10.200.0.0/CL3.kgn.ibm.com 10.200.0.0/CL2.kgn.ibm.com'** attribute.

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

   The **10.200.0.0/CL3.kgn.ibm.com** portion allows all **CL3** nodes to communicate over their Gigabit Ethernet. The **10.200.0.0/CL2.kgn.ibm.com** portion allows remote mounts between clusters **CL3** and **CL2** to communicate over their Gigabit Ethernet.

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

Note: Use of the **subnets** attribute does not ensure a highly available system. If the GPFS daemon is using the IP address specified by the **subnets** attribute, and that interface goes down, GPFS does not switch to the other network. You can use **mmdiag --networks** to verify that the subnet is in fact being used.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 29

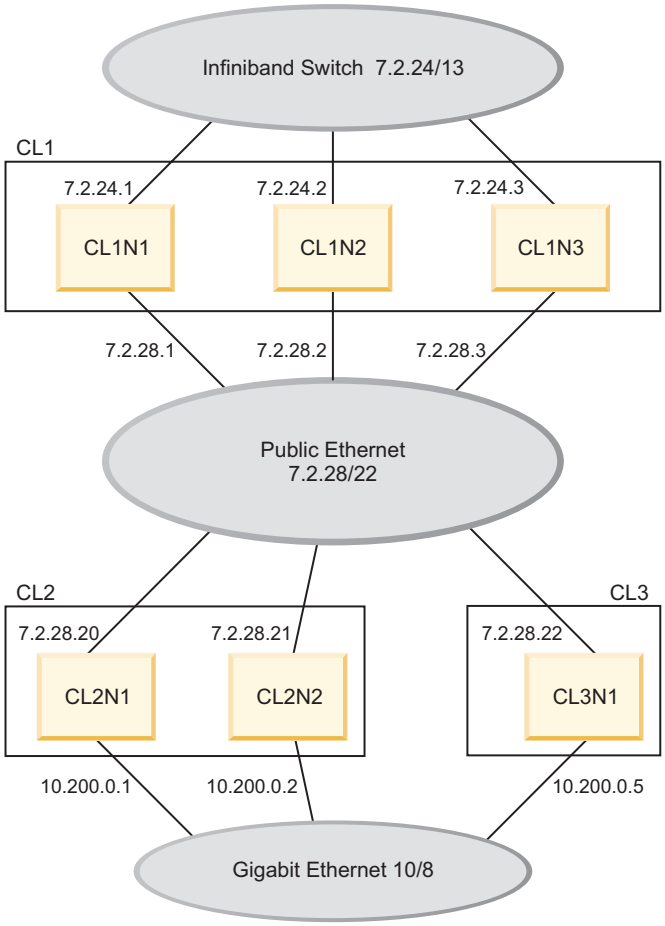


Figure 4. Use of public and private IP addresses in three GPFS clusters

Using multiple security levels

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

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

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

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

   The administrator of **cluster1** issues these commands:

```
mmauth add cluster2 -k keyFile -l AUTHONLY
mmauth add cluster3 -k keyFile -l AES128-SHA
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 30

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

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

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

Changing security keys

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

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

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

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

   ```
   mmauth genkey new
   ```

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

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

   where:

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

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

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

   ```
   mmauth genkey commit
   ```

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 31

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

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

   ```
   mmauth genkey new
   ```

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

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

   where:

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

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

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

   ```
   mmauth genkey commit
   ```

NIST compliance

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

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

The valid values for **nistCompliance** are documented in *GPFS: Administration and Programming*.

Additional information about GPFS file systems accessed by nodes that belong to other GPFS clusters

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 32

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 33

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 34

Chapter 2. Information Lifecycle Management for GPFS

GPFS can help you achieve Information Lifecycle Management (ILM) efficiencies through powerful policy-driven automated tiered storage management. The GPFS ILM toolkit helps you manage sets of files and pools of storage, and also enables you to automate the management of file data.

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

Note: This feature is available with GPFS Standard Edition or higher.

GPFS policy-based ILM tools allow you to:

- Create *storage pools* to provide a way to partition a file system's storage into collections of disks or a redundant array of independent disks (RAIDs) with similar properties that are managed together as a group. GPFS has three types of storage pools:
  - A required **system** storage pool that you create and manage through GPFS
  - Optional user storage pools that you create and manage through GPFS
  - Optional external storage pools that you define with GPFS policy rules and manage through an external application such as Tivoli® Storage Manager
- Create *filesets* to provide a way to partition the file system namespace to allow administrative operations at a finer granularity than that of the entire file system. See "Filesets" on page 52.
- Create *policy rules* based on data attributes to determine initial file data placement and manage file data placement throughout the life of the file. See "Policies and rules" on page 19.

Storage pools

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

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

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

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

- Improved price-performance by matching the cost of storage to the value of the data
- Improved performance by:
  - Reducing the contention for premium storage

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 35

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

For additional information, refer to:

- "Internal GPFS storage pools"
- "External storage pools" on page 18

Internal GPFS storage pools

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

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

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

For additional information, refer to:

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

The system storage pool

The **system** storage pool contains file system control structures, reserved files, directories, symbolic links, special devices, as well as the metadata associated with regular files, including indirect blocks, extended attributes, and so forth. The **system** storage pool can also contain user data. There is only one **system** storage pool per file system, and it is automatically created when the file system is created.

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

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

The system.log storage pool

By default the file system recovery log is stored in the system storage pool with file system metadata. The file system recovery log can also be placed in a dedicated pool that is called the **system.log** pool. This storage pool must be created explicitly. It is highly recommended to only use storage that is as fast or even faster than what is used for the system storage pool. This recommendation is because of the high

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 36

number of small synchronous data updates made to the recovery log. The block size for the **system.log** pool must be the same as the block size of the system pool.

There can be at most one **system.log** storage pool per file system.

User storage pools

All user data for a file is stored in the assigned storage pool as determined by your file placement rules. In addition, file data can be migrated to a different storage pool according to your file management policies. For more information on policies, see "Policies and rules" on page 19.

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

Tracking file access temperature within a storage pool

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

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

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

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

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

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

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

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

Use **WEIGHT(FILE_HEAT)** with a policy **MIGRATE** rule to prioritize migration by file temperature. (You can use the **GROUP POOL** rule to define a group pool to be specified as the **TO POOL** target.) See "Policies and rules" on page 19.

Managing storage pools

Managing your storage pools includes:

- "Creating storage pools" on page 16

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 37

- "Changing the storage pool assignment for a disk"
- "Changing the storage pool assignment for a file"
- "Deleting storage pools"
- "Listing storage pools for a file system" on page 17
- "Listing the storage pool for a file" on page 17
- "Listing disks in a storage pool and associated statistics" on page 17
- "Rebalancing files in a storage pool" on page 18
- "Using replication with storage pools" on page 18

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

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

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

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

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

Changing the storage pool assignment for a disk:
Once a disk is assigned to a storage pool, the pool assignment cannot be changed using either the **mmchdisk** command or the **mmrpldisk** command. To move a disk to another pool:
1. Delete the disk from its current pool by issuing the **mmdeldisk** command. This will move the data to the remaining disks in the storage pool.
2. Add the disk to the new pool by issuing the **mmadddisk** command.
3. Rebalance the data across all disks in the new storage pool by issuing the **mmrestripefs -P** command.

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

By default, both of these commands migrate data immediately (this is the same as using the **-I yes** option for these commands). If desired, you can delay migrating the data by specifying the **-I defer** option for either command. Using the defer option, the existing data does not get moved to the new storage pool until either the **mmrestripefs** command or the **mmrestripefile** command are executed. For additional information, refer to:
- "Policies" on page 19
- "Rebalancing files in a storage pool" on page 18

Deleting storage pools:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 38

Deleting the **system** storage pool is not allowed. In order to delete the **system** storage pool, you must delete the file system.

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

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

For example, this command:

```
mmlsfs fs1 -P
```

produces output similar to this:

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

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

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

For example, this command:

```
mmlsattr -L myfile
```

produces output similar to this:

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

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

Listing disks in a storage pool and associated statistics:
To list the disks belonging to a storage pool, issue the **mmdf -P** command.

For example, this command:

```
mmdf fs1 -P sp1
```

produces output similar to this:

```
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: sp1 (Maximum disk size allowed is 1.2 TB)
vp4vsdn05            17760256        6 no       yes        11310080 ( 64%)        205200 ( 1%)
vp5vsdn05            17760256        6 no       yes        11311104 ( 64%)        205136 ( 1%)
vp6vsdn05            17760256        6 no       yes        11300352 ( 64%)        206816 ( 1%)
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 39

```
vp7vsdn05            17760256        6 no       yes        11296256 ( 64%)        209872 ( 1%)
vp0vsdn05            17760256        6 no       yes        11293696 ( 64%)        207968 ( 1%)
vp1vsdn05            17760256        6 no       yes        11293184 ( 64%)        206464 ( 1%)
vp2vsdn05            17760256        6 no       yes        11309056 ( 64%)        203248 ( 1%)
vp3vsdn05            17760256        6 no       yes        11269120 ( 63%)        211456 ( 1%)
                -------------                         -------------------- -------------------
(pool total)        142082048                              90382848 ( 64%)       1656160 ( 1%)
```

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

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

Rebalancing files in a storage pool:
A root user can rebalance file data across all disks in a file system by issuing the **mmrestripefs** command. Optionally:

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

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

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

External storage pools

When you initially create a file, GPFS assigns that file to an internal storage pool. Internal storage pools support various types of online storage. To move data from online storage to offline or near-line storage, you can use external storage pools. External storage pools use a flexible interface driven by GPFS policy rules that simplify data migration to and from other types of storage such as tape storage. For additional information, refer to "Policies and rules" on page 19.

You can define multiple external storage pools at any time using GPFS policy rules. To move data to an external storage pool, the GPFS policy engine evaluates the rules that determine which files qualify for transfer to the external pool. From that information, GPFS provides a list of candidate files and executes the script specified in the rule that defines the external pool. That executable script is the interface to the external application, such as Tivoli Storage Manager (TSM), that does the actual migration of data into an external pool. Using the external pool interface, GPFS gives you the ability to manage information by allowing you to:

1. Move files and their extended attributes onto low-cost near-line or offline storage when demand for the files diminishes.
2. Recall the files, with all of their previous access information, onto online storage whenever the files are needed.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 40

External pool requirements

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

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

For additional information, refer to "Working with external storage pools" on page 45.

Policies and rules

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

GPFS supports these policies:

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

Policies

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

- Initial file placement
- File management
- Restoring file data
- Encryption-specific uses; see Chapter 15, "Encryption," on page 261.

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

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

The management policy determines file management operations such as migration and deletion.

In order to migrate or delete data, you must use the **mmapplypolicy** command. You can define the file management rules and install them in the file system together with the placement rules. As an

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 41

alternative, you may define these rules in a separate file and explicitly provide them to **mmapplypolicy** using the **-P** option. In either case, policy rules for placement or migration may be intermixed. Over the life of the file, data can be migrated to a different storage pool any number of times, and files can be deleted or restored.

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

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

GPFS performs error checking for file-placement policies in the following phases:

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

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

```
mmchpolicy Device DEFAULT
```

For more information on using GPFS commands to manage policies, see "Managing policies" on page 44.

Policy rules

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

Rules specify conditions that, when true, cause the rule to be applied. Conditions that cause GPFS to apply a rule include:

- Date and time when the rule is evaluated, that is, the current date and time
- Date and time when the file was last accessed
- Date and time when the file was last modified
- Fileset name
- File name or extension
- File size
- User ID and group ID

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

There are eight types of policy rules that allow you to define specific actions that GPFS will implement on the file data. Each rule has clauses that control candidate selection, namely when the rule is allowed

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 42

to match a file, what files it will match, the order to operate on the matching files and additional attributes to show for each candidate file. Different clauses are permitted on different rules based upon the semantics of the rule.

The rules and their respective syntax diagrams are:

- File placement rules

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

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

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

- File migration rule

```
RULE ['RuleName'] [WHEN TimeBooleanExpression]
  MIGRATE
    [FROM POOL 'FromPoolName']
    [THRESHOLD (HighPercentage[,LowPercentage[,PremigratePercentage]])]
    [WEIGHT (WeightExpression)]
  TO POOL 'ToPoolName'
    [LIMIT (OccupancyPercentage)]
    [REPLICATE (DataReplication)]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [SHOW (['String'] SqlExpression)]
    [SIZE (numeric-sql-expression)]
    [WHERE SqlExpression]
```

- File deletion rule

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

- File exclusion rule

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

- File list rule

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


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 43

- File restore rule

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

- External storage pool definition rule

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

- External list definition rule

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

Policy rule syntax definitions

The GPFS policy rules follow these syntax definitions:

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

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

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

  For example:

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

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

  The GPFS `ESCAPE` clause allows you to add to the set of "unreserved" characters.

  For example:

```
ESCAPE '%/+@#'
```

  specifies that none of the characters in `/+@#` are escaped, so that a path name like `/root/directory/@abc+def#ghi.jkl` will appear in a file list with no escape sequences; whereas under `ESCAPE '%'`, specifying a rigorous RFC3986 encoding yields `%2Froot%2Fdirectory%2F%40abc%2Bdef%23ghi.jkl`.

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

EXCLUDE
: Identifies a file exclusion rule.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 44

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

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

EXEC '*InterfaceScript***'**
Specifies an external program to be invoked to pass requests to an external storage management application. *InterfaceScript* should be a fully-qualified pathname to a user-provided script or program that supports the commands described in "User provided program for managing external pools" on page 46.

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

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

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

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

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

Optionally, a **LIMIT**, specified as an occupancy percentage, may be specified for each disk pool; if not specified, the limit defaults to 99%. The **THEN** keyword signifies that the disk pools specified before a **THEN** keyword are preferred over those specified after. When a pool defined by a **GROUP POOL** rule is the **TO POOL** target of a **MIGRATE** rule, the selected files will be distributed among the disk pools comprising the group pool, with files of highest weight going to the most preferred disk pool up to the occupancy limit for that pool. If there are still more files to be migrated, those will go to the second most preferred pool up to the occupancy limit for that pool (again choosing the highest-weight files from among the remaining selected files).

A pool defined by a **GROUP POOL** rule may be specified in a **FROM POOL** clause, in which case the **FROM POOL** clause matches any file in any of the disk pools listed in the **GROUP POOL**.

You may "re-pack" a group pool by **WEIGHT**, migrating files of higher weight to preferred disk pools by specifying a group pool as both the source and the target of a **MIGRATE** rule. For example:

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

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

LIMIT(*OccupancyPercentage***)**
Used to limit the creation of data in a storage pool. If it is determined that transferring the file to the specified pool would exceed the specified occupancy percentage for that pool, GPFS skips the rule and the policy engine looks for the next rule that seems to match. See "Phase two - choosing and scheduling files" on page 37.

OccupancyPercentage can be specified as a floating point number. For example:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 45

```
rule 'r' restore to pool 'x' limit(8.9e1)
```

For testing or planning purposes, and when using the **mmapplypolicy** command with the **-I defer** or **-I test** options, it is acceptable to specify a **LIMIT** larger than 100%.

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

LIST *ListName*
Identifies a file list generation rule. A given file may match more than one list rule but will be included in a given list only once. *ListName* provides the binding to an **EXTERNAL LIST** rule that specifies the executable program to use when processing the generated list.

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

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

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

RESTORE TO POOL *PoolName*
Identifies a file restore rule. When a file is restored using the **gpfs_fputattrswithpathname()** subroutine, this rule allows you to match files against their saved attributes rather than the current file attributes.

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

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

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

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

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

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

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

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

THRESHOLD (*HighPercentage***[,***LowPercentage***[,***PremigratePercentage***]])**
Used to control migration and deletion based on the pool storage utilization (percent of assigned storage occupied).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 46

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

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

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

Notes:

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

THRESHOLD (*Resource_class***)**
Specifies which kind of capacity-managed resources are associated with *ListName*. Supported values are:

FILESET_QUOTAS
Indicates that the LIST rule will use the occupancy percentage of fileset quotas when using the threshold rule.

GROUP_QUOTAS
Indicates that the LIST rule will use the occupancy percentage of group quotas when using the threshold rule.

POOL_CAPACITIES
If threshold is not specified in the EXTERNAL LIST rule but used LIST rule, the resource class will be POOL_CAPACITIES, by default. This indicates that the LIST rule will use the occupancy percentage of pool when using the threshold rule. This is the default.

USER_QUOTAS
Indicates that the LIST rule will use the occupancy percentage of user quotas when using the threshold rule.

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

For more detail on how **THRESHOLD** can be used to control file migration and deletion, see "Phase one - selecting candidate files" on page 36 and "Pre-migrating files with external storage pools" on page 49.

TO POOL *ToPoolName*
Specifies the name of the storage pool where all files that match the rule criteria will be migrated.

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 47

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

The **mmapplypolicy** command assigns the **CURRENT_TIMESTAMP** once when it begins processing, using either the actual UTC date and time or the date specified with the **-D** option.

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

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

You can omit the **WHERE** clause entirely. This is equivalent to writing **WHERE TRUE**. When used, the **WHERE** clause must be the last clause of the rule.

SQL expressions for policy rules

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

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

ACCESS_TIME
Specifies an SQL timestamp value for the date and time that the file was last accessed (POSIX atime).

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

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

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

CLONE_IS_PARENT
Specifies if the file is a clone parent.

CLONE_PARENT_FILESET_ID
Specifies the fileset ID of the clone parent. The fileset ID is only available if **CLONE_PARENT_IS_SNAP** is a non-zero value.

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

CLONE_PARENT_IS_SNAP
Specifies if the clone parent is in a snapshot.

CLONE_PARENT_SNAP_ID
Specifies the snapshot ID of the clone parent. The snapshot ID is only available if **CLONE_PARENT_IS_SNAP** is a non-zero value.

CREATION_TIME
Specifies an SQL timestamp value that is assigned when a file is created.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 48

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

DIRECTORY_HASH
May be used to group together files within the same directory.

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

DIRECTORY_HASH uses the following functions:

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

HashToFloat(*StringValue***)**
Is a hash function that returns a quasi-random floating point number >=0 and <1, whose value depends on string-value. Although the result may appear random, for any given string value **HashToFloat**(*StringValue*) always returns the same floating point value.

This rule:

```
RULE 'y' LIST 'xl' SHOW(DIRECTORY_HASH)
```

results in:

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

This rule:

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

causes files within the same directory to be grouped and processed together during deletion, which may improve the performance of GPFS directory locking and caching.

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

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

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

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

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

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

INODE
Specifies the file's inode number.

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

MODE
Represents the file type and permission bits as a string in a format that mimics the first 10 characters of the output of each file descriptive line of the Unix command **ls -l**.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 49

For example, **-rwxr-xr-x** is the **MODE** string of a file that may be read or executed by its owner, group, or any user, but only written by its owner. The first character of **MODE** represents the file type; GPFS supports the following:

- data

d directory

l link

c character device

b block device

p pipe

s socket

? unknown or something else

MISC_ATTRIBUTES

Specifies a variety of miscellaneous file attributes. The value is a concatenated string of attributes that are defined as:

a appendOnly

A archive

c compressed

D directory (to match all directories, you can use %D% as a wildcard)

e encrypted

Note: This does not refer to GPFS encryption.

F regular data file

H hidden

i not indexed by content

L symbolic link

o offline

O other (not F, D, nor L) for example, maybe a device or named pipe

M co-managed

2 data blocks are replicated

I some data blocks may be ill-placed

J some data blocks may be ill-replicated

p reparse point

P AFM

r has streams

R read-only

s sparse file

S system

t temporary

u cached complete

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 50

U **trunc**-managed

V read-managed

W write-managed

X immutability

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

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

NAME
Specifies the name of a file.

NLINK
Specifies the number of hard links to the file.

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

POOL_NAME
Specifies the storage pool where the file data resides.

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

RDEVICE_ID
Specifies the device type for a device.

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

Notes:

1. When file attributes are referenced in initial placement rules, only the following attributes are valid: **NAME**, **USER_ID**, **GROUP_ID**, and **FILESET_NAME**. The placement rules, like all rules with a WHERE clause, may also reference the current date and current time and use them to control matching.
2. When file attributes are used for restoring files, the attributes that are used for a file correspond to the file's attributes at the time of its backup, not to the current restored file.
3. For SQL expressions, if you wish to show any of these attribute fields as strings (for example, FILE_HEAT), use SHOW('[FILE_HEAT]') rather than SHOW('FILE_HEAT'), as the latter will be expanded.

Using built-in functions:
With GPFS, you can use built-in functions in comparison predicates, between predicates, in predicates, like predicates, mathematical value expressions, and boolean, string and numeric literals. These functions are organized into the following categories:

- "Extended attribute functions" on page 30
- "String functions" on page 33
- "Numerical functions" on page 34
- "Date and time functions" on page 34

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 51

Extended attribute functions:

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

SetBGF(*BlockGroupFactor***)**
Specifies how many file system blocks are laid out sequentially on disk to behave like a single large block. This option only works if **--allow-write-affinity** is set for the data pool.

SetWAD(*WriteAffinityDepth***)**
Specifies the allocation policy to be used. This option only works if **--allow-write-affinity** is set for the data pool.

SetWADFG("*WadfgValueString***")**
Indicates the range of nodes (in a shared nothing architecture) where replicas of blocks in the file are to be written. You use this parameter to determine the layout of a file in the cluster so as to optimize the typical access patterns of your applications.

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

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

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

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

For example, the following value

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

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

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

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

Notes:

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

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

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

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

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

This rule has the same effect as the following command:

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 52

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

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

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

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

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

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

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

```
define(DISPLAY_NULL,[CASE WHEN ($1) IS NULL THEN '_NULL_' ELSE ($1) END])
rule external list 'a' exec ''
rule list 'a' SHOW(DISPLAY_NULL(xattr('user.marc')) || ' and ' || DISPLAY_NULL(xattr('user.eric')))
```

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

```
[root@wolf12 cmvc]# mmapplypolicy /gig/sill -P /ghome/makaplan/policies/display-null.policy -I test -L 2
   ...
WEIGHT(inf) LIST 'a' /gg/sll/cc SHOW(_NULL_ and _NULL_)
WEIGHT(inf) LIST 'a' /gg/sll/mm SHOW(yes-marc and _NULL_)
WEIGHT(inf) LIST 'a' /gg/sll/bb SHOW(_NULL_ and yes-eric)
WEIGHT(inf) LIST 'a' /gg/sll/tt SHOW(yes-marc and yes-eric)
```

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 53

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

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

XATTR_FLOAT(*extended-attribute-name* **[,** *start* **[,** *length***, [,** *conversion_option* **]]])**

Returns the value of a substring of the extended attribute that is named by its argument, converted to an SQL double floating-point value, where:

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

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

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

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

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

Notes:

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

XATTR_INTEGER(*extended-attribute-name* **[,** *start* **[,** *length***, [,** *conversion_option* **]]])**

Returns the value of (a substring of) the extended attribute named by its argument, converted to a SQL LARGEINT value, where.

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

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 54

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

Notes:

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

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

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

Important tips:

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

CHAR(*expr***[,** *length***])**

Returns a fixed-length character string representation of its *expr* argument, where:

expr Can be any data type.

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

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

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

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

CONCAT(*x***,***y***)**

Concatenates strings *x* and *y*.

HEX(*x***)**

Converts an integer *x* into hexadecimal format.

LENGTH(*x***)**

Determines the length of the data type of string *x*.

LOWER(*x***)**

Converts string *x* into lowercase.

REGEX(*String***,'***Pattern***')**

Returns TRUE if the pattern matches, FALSE if it does not. *Pattern* is a Posix extended regular expression.

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

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

The following SQL expression:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 55

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

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

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

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

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

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

expr Can be any data type.

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

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

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

Numerical functions: You can use these numeric-calculation functions to place files based on either numeric parts of the file name, numeric parts of the current date, UNIX-client user IDs or group IDs. These can be used in combination with comparison predicates and mathematical infix operators (such as addition, subtraction, multiplication, division, modulo division, and exponentiation).

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

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

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

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

CURRENT_DATE
Determines the current date on the GPFS server.

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 56

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example of a policy rules file

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

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

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


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 57

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

Miscellaneous SQL functions: The following miscellaneous SQL functions are available:

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

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

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

Semantics of the mmapplypolicy command and its policy rules

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

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

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

1. "Phase one - selecting candidate files"
2. "Phase two - choosing and scheduling files" on page 37
3. "Phase three - migrating and deleting files" on page 39

Phase one - selecting candidate files

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

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

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

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

- If the rule has a **WHEN** clause that evaluates to **FALSE**, the rule is skipped.
- If the rule has a **FROM POOL** clause, and the named pool does not match the **POOL_NAME** attribute of the file, the rule is skipped. A **FROM POOL** clause that specifies a group pool name matches a file if any pool name within the group pool matches the **POOL_NAME** attribute of the file.
- If there is a **THRESHOLD** clause and the current pool of the file has an occupancy percentage that is less than the *HighPercentage* parameter of the **THRESHOLD** clause, the rule is skipped.
- If the rule has a **FOR FILESET** clause, but none of the named filesets match the **FILESET_NAME** attribute of the file, the rule is skipped.
- If the rule has a **WHERE** clause that evaluates to **FALSE**, the rule is skipped. Otherwise, the rule applies.
- If the applicable rule is a **LIST** *'listname-y'* rule, the file becomes a candidate for inclusion in the named list unless the **EXCLUDE** keyword is present, in which case the file will not be a candidate; nor will any following **LIST** *'listname-y'* rules be considered for the subject file. However, the file is subject to **LIST** rules naming other list names.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 58

- If the applicable rule is an **EXCLUDE** rule, the file will be neither migrated nor deleted. Files matching the **EXCLUDE** rule are not candidates for any **MIGRATE** or **DELETE** rule.

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

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

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

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

  When a group pool is the **TO POOL** target of a **MIGRATE** rule, the selected files are distributed among the disk pools comprising the group pool, with files of highest weight going to the most preferred disk pool up to the occupancy limit for that pool. If there are still more files to be migrated, those go to the second most-preferred pool up to the occupancy limit for that pool (again choosing the highest-weight files from among the remaining selected files); and so on for the subsequent most-preferred pools, until either all selected files have been migrated or until all the disk pools of the group pool have been filled to their respective limits.
- If the applicable rule is a **DELETE** rule, the file becomes a *candidate* for deletion.
- If there is no applicable rule, the file is not a candidate for migration or deletion.
- Each candidate file (for migration or deletion) is also associated with a *LowPercentage* occupancy percentage value, which is taken from the **THRESHOLD** clause of the applicable rule. If not specified, the *LowPercentage* value defaults to 0%.
- Each candidate file is also associated with a numeric *weight*, either computed from the *WeightExpression* of the applicable rule, or assigned a default using these rules:
  - If a *LowPercentage* is specified within a **THRESHOLD** clause of the applicable rule, the weight of the candidate is taken as the **KB_ALLOCATED** attribute of the candidate file.
  - If a *LowPercentage* is not specified within a **THRESHOLD** clause of the applicable rule, the weight of the candidate is taken as **+infinity**.

Phase two - choosing and scheduling files

In the second phase of the **mmapplypolicy** job, some or all of the candidate files are chosen. Chosen files are scheduled for migration or deletion, taking into account the weights and thresholds determined in "Phase one - selecting candidate files" on page 36, as well as the actual pool occupancy percentages. Generally, candidates with higher weights are chosen ahead of those with lower weights.

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

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

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 59

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

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

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

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

File grouping and the SIZE clause

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

However, you can set up **mmapplypolicy** to schedule files so that each invocation of the InterfaceScript gets approximately the same amount of file data to process. To do so, use the **SIZE** clause of certain policy rules to specify that scheduling be based on the sum of the sizes of the files. The **SIZE** clause can be applied to the following rules (for details, see "Policy rules" on page 20):

- **DELETE**
- **EXTERNAL LIST**
- **EXTERNAL POOL**
- **LIST**
- **MIGRATE**

Administrator-specified customized file grouping or aggregation

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

You can accomplish this by following these steps:

1. Run **mmapplypolicy** with the **-I prepare** option to produce a list of chosen candidate files, but not pass the list to a InterfaceScript.
2. Use your aggregator program to sort the list of chosen candidate files into groups with similar attributes and write each group to a new, separate file list.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 60

3. Run **mmapplypolicy** with the **-r** option, specifying a set of file list files to be read. When invoked with the **-r** option, **mmapplypolicy** does not choose candidate files; rather, it passes the specified file lists as input to the InterfaceScript.

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

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

Reasons for candidates not to be chosen for deletion or migration

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

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

Phase three - migrating and deleting files

In the third phase of the **mmapplypolicy** job, the candidate files that were chosen and scheduled by the second phase are migrated or deleted, each according to its applicable rule. For migrations, if the applicable rule had a **REPLICATE** clause, the replication factors are also adjusted accordingly. It is acceptable for the effective **FROM POOL** and **TO POOL** to be the same because the **mmapplypolicy** command can be used to adjust the replication factors of files without necessarily moving them from one pool to another.

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

Policy rules - examples and tips

Before you write and apply policies, consider this advice:

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

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

Then run the following command:

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

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

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


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 61

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

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

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

Note: For examples and more information on the **-L** flag, see the section: "The **mmapplypolicy -L** command" in the *GPFS: Problem Determination Guide*.

The following examples and tips are provided to illustrate how to perform simple tasks using policy rules with files controlled by GPFS file systems:

1. Each rule begins with the keyword **RULE** and is followed by the name of the rule. Although the rule name is optional, it should be provided and should be unique. Providing a rule name and keeping it unique will help you match error messages with faulty rules.
2. If the storage pool named **pool_1** has an occupancy percentage above 90% now, bring the occupancy percentage of **pool_1** down to 70% by migrating the largest files to storage pool **pool_2**:

   ```
   RULE 'mig1' MIGRATE FROM POOL 'pool_1'
           THRESHOLD(90,70) WEIGHT(KB_ALLOCATED) TO POOL 'pool_2'
   ```
3. Delete files from the storage pool named **pool_1** that have not been accessed in the last 30 days, and are named like temporary files or appear in any directory that is named **tmp**:

   ```
   RULE 'del1' DELETE FROM POOL 'pool_1'
      WHERE (DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 30)
           AND (lower(NAME) LIKE '%.tmp' OR PATH_NAME LIKE '%/tmp/%')
   ```
4. Use the SQL **LIKE** predicate to test file names and path names:

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

   Where:

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

   Use the optional **ESCAPE** clause to establish an escape character, when you need to match '_' or '%' exactly.
5. Use the SQL **UPPER** and **LOWER** functions to ignore case when testing names:

   ```
   RULE 'UPPER'  DELETE WHERE upper(PATH_NAME) LIKE '%/TMP/OLD/%'
   RULE 'lower'  DELETE WHERE lower(PATH_NAME) LIKE '%/tmp/old/%'
   ```
6. Use the SQL **SUBSTR** or **SUBSTRING** functions to test a substring of a name:

   ```
   RULE 's1' DELETE WHERE SUBSTRING(NAME FROM 1 FOR 5)='XXXX-'
   RULE 's2' DELETE WHERE SUBSTR(NAME,1,5)='YYYY-'
   ```
7. Use the SQL **SUBSTR** and **LENGTH** functions to test the suffix of a name:

   ```
   RULE 'sfx' DELETE WHERE SUBSTR(NAME,LENGTH(NAME)-3)='.tmp'
   ```
8. Use a **WHEN** clause to restrict rule applicability to a particular day of the week:

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

   CURRENT_DATE is an SQL built in operand that returns the date portion of the **CURRENT_TIMESTAMP** value.
9. Use the SQL **IN** operator to test several possibilities:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 62

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

For information on how to use a macro processor such as **m4** to make reading and writing policy rules easier, see "Using macro processing utilities to simplify policy creation, comprehension, and maintenance" on page 43.

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

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

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

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

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

Notes:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RULE '3month-old' MIGRATE FROM POOL 'pool_X'
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 63

```
THRESHOLD(90,80) WEIGHT(weight_expression)
TO POOL 'pool_ZZ'
WHERE  DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 90
```

More information about weights is available in the next example.

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

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

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

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

where:

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

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

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

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

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

This means:

- Give a very large weight bias to any file older than a year.
- Force the weight of any file younger than 30 days to 0.
- Assign weights to all other files according to the number of kilobytes occupied.

The formal SQL syntax is this:

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

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

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

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

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 64

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

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

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

The following is also supported:

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

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

Using macro processing utilities to simplify policy creation, comprehension, and maintenance

Prior to evaluating the policy rules, GPFS invokes the **m4** macro processor to process the policy file. This allows you to incorporate into the policy file some of the traditional **m4** facilities and to define simple and parameterized macros, conditionally include text, perform conditional evaluation, perform simple string operations, perform simple integer arithmetic and much more.

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

Utilizing **m4** as a front-end processor simplifies writing policies and produces policies that are easier to understand and maintain. Here is Example 16 on page 42 from "Policy rules - examples and tips" on page 39 written with a few **m4** style macro definitions:

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

define(weight_expression,
       CASE
        WHEN access_age > 365
          THEN 100000 + access_age
        WHEN access_age < 30
          THEN 0
        ELSE
          KB_ALLOCATED
        END
)

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

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

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

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

RULE 'gb1' WHEN(DAY_OF_WEEK IN (SATURDAY,SUNDAY))
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 65

```
        MIGRATE TO POOL 'ypooly' WHERE GB_ALLOCATED >= .015

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

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

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

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

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

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

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

Managing policies

Policies and the rules that they contain are used to assign files to specific storage pools. A storage pool typically contains a set of volumes that provide a specific quality of service for a specific use, such as to store all files for a particular application or a specific business division.

Managing policies includes:

- "Creating a policy"
- "Installing a policy"
- "Changing the active policy" on page 45
- "Listing policies" on page 45
- "Validating policies" on page 45
- "Deleting policies" on page 45

Creating a policy

Create a text file for your policy following these guidelines:

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

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

  If you do not do this, and no other rule applies, an **EINVAL** error code is returned.
- Comments within a policy must start with a **/*** and end with a ***/**:

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

See "Policy rules" on page 20

Installing a policy

To install a policy:

1. Create a text file containing the desired policy rules.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 66

2. Issue the **mmchpolicy** command.

Changing the active policy

Prepare a file with the new or changed policy rules and then issue the **mmchpolicy** command. The **mmchpolicy** command will activate the following sequence of events:

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

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

Listing policies

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

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

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

Validating policies

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

Deleting policies

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

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

Working with external storage pools

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

- Defining the external pools
- "User provided program for managing external pools" on page 46
- "File list format" on page 47
- "Record format" on page 47
- "Migrate and recall with external pools" on page 48
- "Pre-migrating files with external storage pools" on page 49
- "Purging files from external storage pools" on page 49
- "Using thresholds with external pools" on page 50

Defining external pools

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 67

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

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

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

Where:

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

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

Example of a rule that defines a storage pool

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

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

In this example:

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

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

User provided program for managing external pools

Once you have defined an external storage pool, subsequent migration or deletion rules may refer to that pool as a source or target storage pool. When the **mmapplypolicy** command is invoked and a rule dictates that data should be moved to or from an external pool, the user provided program identified with the **EXEC** clause in the policy rule launches. That executable program receives three arguments:

- The command to be executed. Your script should implement each of the following sub-commands:
  - LIST - Provides arbitrary lists of files with no semantics on the operation.
  - MIGRATE - Migrate files to external storage and reclaim the online space allocated to the file.
  - PREMIGRATE - Migrate files to external storage but do not reclaim the online space.
  - PURGE - Delete files from both the online file system and the external storage.
  - RECALL - Recall files from external storage to the online storage.
  - TEST – Test for presence and operation readiness. Return zero for success. Return non-zero if the script should not be used on a given node.
- The name of a file containing a list of files to be migrated, premigrated, or purged. See "File list format" on page 47 for detailed description of the layout of the file.
- Any optional parameters specified with the OPTS clause in the rule. These optional parameters are not interpreted by the GPFS policy engine.

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 68

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

File list format

Each call to the external pool script specifies the pathname for a temporary file that contains a list of files to be operated on. This file list defines one file per line as follows:

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

where:

InodeNumber is a 64-bit inode number.

GenNumber is a 32-bit file generation number.

SnapId is a 64-bit snapshot identifier.

OptionalShowArgs is the result, if any, from the evaluation of the SHOW clause in the policy rule.

FullPathToFile is a fully qualified path name to the file. When there are multiple paths within a file system to a particular file (*Inode*, *GenNumber*, and *SnapId*), each path is shown.

The "--" characters are a field delimiter that separates the optional show parameters from the path name to the file.

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

Record format

The format of the records in each file list file can be expressed as:

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

where:

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 69

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

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

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

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

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

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

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

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

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

For more information about the **mmapplypolicy** command, refer to the *GPFS: Administration and Programming Reference*.

Migrate and recall with external pools

Once you have defined an external storage pool, subsequent migration or deletion rules may refer to that pool as a source or target storage pool. When you invoke **mmapplypolicy** and a rule dictates that data should be deleted or moved to or from an external pool, the program identified in the EXTERNAL POOL rule is invoked with the following arguments:

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

For example, let us assume an external pool definition:

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

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

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

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

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


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 70

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

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

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

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

Notes:

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

   ```
   RULE 'exclude hsm system files' EXCLUDE WHERE PATH_NAME LIKE '%/.SpaceMan%'
   ```

Pre-migrating files with external storage pools

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

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

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

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

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

Note:

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

Purging files from external storage pools

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 71

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

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

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

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

Using thresholds with external pools

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

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

A NO_SPACE event is generated if the file system is out of space. A LOW_SPACE event requires a threshold. If a threshold is specified, a LOW_SPACE event will be generated.

GPFS provides user exits for NO_SPACE and LOW_SPACE events. Using the **mmaddcallback** command, you can specify a script that will run when either of these events occurs. For more information about the **mmaddcallback** command, see the *GPFS: Administration and Programming Reference*.

The file with the policy rules used by **mmapplypolicy** is the one that is currently installed in the file system. The HSM user should define migration or deletion rules to reduce the usage in each online storage pool. Migration rules defined with a high and low THRESHOLD establish the threshold used to signal the LOW_SPACE event for that pool. Because more than one migration rule can be defined, the threshold for a pool is the minimum of the high thresholds set by the rules for that pool. Each pool has its own threshold. Pools without migration rules do not signal a LOW_SPACE event.

Backup and restore with storage pools

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

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

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

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 72

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

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

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

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

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

The **gpfs_fgetattrs()**, **gpfs_fputattrs()** and **gpfs_fputattrswithpathname()** subroutines have several optional flags that you can use to further control the storage pools selection. For detailed information, refer to the *GPFS: Administration and Programming Reference*.

Working with external lists

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

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

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

Where:

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

See "User provided program for managing external pools" on page 46.

Example

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

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

In this example:

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 73

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

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

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

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

Filesets

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

In addition, GPFS utilizes a file system object called a *fileset*. A fileset is a subtree of a file system namespace that in many respects behaves like an independent file system. Filesets provide a means of partitioning the file system to allow administrative operations at a finer granularity than the entire file system:

- Filesets can be used to define quotas on both data blocks and inodes.
- The owning fileset is an attribute of each file and can be specified in a policy to control initial data placement, migration, and replication of the file's data. See “Policies and rules” on page 19.
- Fileset snapshots can be created instead of creating a snapshot of an entire file system.

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

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

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

Fileset namespace

A newly created fileset consists of an empty directory for the root of the fileset, and it is initially not linked into the file system's namespace. A newly created fileset is not visible to the user until it is attached to the namespace by issuing the **mmlinkfileset** command. Filesets are attached to the namespace with a special link called a *junction*. A junction is a special directory entry, much like a POSIX hard link, that connects a name in a directory of one fileset to the root directory of another fileset. Only one junction is allowed per fileset, so that a fileset has a unique position in the namespace and a unique path to any of its directories. The target of the junction is referred to as the *child fileset*, and a fileset can have any number of children. From the user's viewpoint, a junction always appears as if it were a directory, but the user is not allowed to issue the **unlink** or **rmdir** commands on a junction.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 74

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

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

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

See "Managing filesets" on page 55.

Filesets and quotas

The GPFS quota commands support the **-j** option for fileset block and inode allocation. The quota limit on blocks and inodes in a fileset are independent of the limits for specific users or groups of users. See these commands:

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

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

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

Filesets and storage pools

Filesets are not specifically related to storage pools, although each file in a fileset physically resides in blocks in a storage pool. This relationship is many-to-many; each file in the fileset can be stored in a different user storage pool. A storage pool can contain files from many filesets. However, all of the data for a particular file is wholly contained within one storage pool.

Using file-placement policies, you can specify that all files created in a particular fileset are to be stored in a specific storage pool. Using file-management policies, you can define how files in a specific fileset are to be moved or deleted during the file's life cycle. See "Policy rule syntax definitions" on page 22.

Filesets and global snapshots

A GPFS global snapshot preserves the contents of the entire file system, including all its filesets, even unlinked ones. The state of filesets in the snapshot is unaffected by changes made to filesets in the active

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 75

file system, such as unlink, link or delete. The saved file system can be accessed through the **.snapshots** directories and the namespace, including all linked filesets, appears as it did when the snapshot was created. Unlinked filesets are inaccessible in the snapshot, as they were in the active file system. However, restoring a snapshot also restores the unlinked filesets, which can then be re-linked and accessed.

If a fileset is included in a global snapshot, it can be deleted but it is not entirely removed from the file system. In this case, the fileset is emptied of all contents and given a status of 'deleted'. The contents of a fileset remain available in the snapshots that include the fileset (that is, through some path containing a **.snapshots** component) even after the fileset is deleted, since all the contents of the fileset are saved when a snapshot is created. The fileset remains in the deleted state until the last snapshot containing it is deleted, at which time the fileset is automatically deleted.

A fileset is included in a global snapshot if the snapshot is created after the fileset was created. Deleted filesets appear in the output of the **mmlsfileset** and **mmlsfileset --deleted** commands, and the **-L** option can be used to display the latest snapshot that includes a fileset.

During a restore from a global snapshot, attributes of filesets included in the snapshot can be altered. The filesets included in the global snapshot are restored to their former state, and newer filesets are deleted. Also, restore may undelete deleted filesets and change linked filesets to unlinked or vice versa. If the name of a fileset was changed since the snapshot was taken, the old fileset name will be restored.

Fileset-level snapshots

Instead of creating a global snapshot of an entire file system, a fileset snapshot can be created to preserve the contents of a single independent fileset plus all dependent filesets that share the same inode space. If an independent fileset has dependent filesets that share its inode space, then a snapshot of the independent fileset will also include those dependent filesets. In other words, a fileset snapshot is a snapshot of the whole inode space.

Each independent fileset has its own hidden **.snapshots** directory in the root directory of the fileset that contains any fileset snapshots. The **mmsnapdir** command allows setting an option that makes global snapshots also available through **.snapshots** in the root directory of all independent filesets. The **.snapshots** directory in the file system root directory lists both global snapshots and fileset snapshots of the root fileset (the root fileset is an independent fileset). This behavior can be customized with the **mmsnapdir** command.

Fileset snapshot names need not be unique across different filesets, so it is valid to use the same name for fileset snapshots of two different filesets because they will appear under **.snapshots** in two different fileset root directories.

| Independent fileset snapshot data and attribute files can be restored using the **mmrestorefs** command.
| See the **mmrestorefs** command in the *GPFS: Administration and Programming Reference* for complete usage
| information.

Filesets and backup

The **mmbackup** command and TSM are unaware of the existence of filesets. When restoring a file system that had been backed up to TSM, the files are restored to their original path names, regardless of the filesets of which they were originally a part.

TSM has no mechanism to create or link filesets during restore. Therefore, if a file system is migrated to TSM and then filesets are unlinked or deleted, restore or recall of the file system does not restore the filesets.

During a full restore from backup, all fileset information is lost and all files are restored into the root fileset. It is recommended that you save the output of the **mmlsfileset** command to aid in the

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 76

reconstruction of fileset names and junction locations. Saving **mmlsfileset -L** also allows reconstruction of fileset comments. Both command outputs are needed to fully restore the fileset configuration.

A partial restore can also lead to confusion if filesets have been deleted, unlinked, or their junctions moved, since the backup was made. For example, if the backed up data was in a fileset that has since been unlinked, the restore process puts it into files and directories in the parent fileset. The unlinked fileset cannot be re-linked into the same location until the restored data is moved out of the way. Similarly, if the fileset was deleted, restoring its contents does not recreate the deleted fileset, but the contents are instead restored into the parent fileset.

Since the **mmbackup** command operates by traversing the directory structure, it does not include the contents of unlinked filesets, even though they are part of the file system. If it is desired to include these filesets in the backup, they should be re-linked, perhaps into a temporary location. Conversely, temporarily unlinking a fileset is a convenient mechanism to exclude it from a backup.

Note: It is recommended not to unlink filesets when doing backups. Unlinking a fileset during an **mmbackup** run can cause the following:

- failure to back up changes in files that belong to an unlinked fileset
- expiration of files that were backed up in a previous **mmbackup** run

In summary, fileset information should be saved by periodically recording **mmlsfileset** output somewhere in the file system, where it is preserved as part of the backup process. During restore, care should be exercised when changes in the fileset structure have occurred since the backup was created.

Attention: If you are using the TSM Backup-Archive client you must use caution when you unlink filesets that contain data backed up by TSM. TSM tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to TSM that you deleted the contents of the fileset. Therefore, the TSM Backup-Archive client inactivates the data on the TSM server which may result in the loss of backup data during the expiration process.

Managing filesets

Managing your filesets includes:

- "Creating a fileset"
- "Deleting a fileset" on page 56
- "Linking a fileset" on page 56
- "Unlinking a fileset" on page 57
- "Changing fileset attributes" on page 57
- "Displaying fileset information" on page 57

Creating a fileset

Filesets are created with the **mmcrfileset** command. By default, filesets are created as dependent filesets that share the root's inode space. The **--inode-space** *ExistingFileset* option can be used to create a dependent fileset that will share inode space an existing fileset. The *ExistingFileset* specified can be **root** or any other independent fileset, but cannot be the reserved keyword **new**. The **--inode-space new** option can be used to create an independent fileset with its own dedicated inode space.

A newly created fileset consists of an empty directory for the root of the fileset and it is initially not linked into the existing namespace. Consequently, a new fileset is not visible, nor can files be added to it, but the fileset name is valid and the administrator can establish quotas on it, or policies for it. The administrator must link the fileset into its desired location in the file system's namespace by issuing the **mmlinkfileset** command in order to make use of it.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 77

After the fileset is linked, the administrator can change the ownership and permissions for the new fileset's root directory, which default to **root** and 0700, to allow users access to it. Files and directories copied into, or created within, the fileset's directory will become part of the new fileset.

Fileset names must follow these conventions:

- Are character strings and must be less than 256 characters in length.
- Must be unique within a file system.
- The name **root** is reserved for the fileset of the files system's root directory.

See the **mmcrfileset** and **mmlinkfileset** commands in the *GPFS: Administration and Programming Reference* for complete usage information.

Deleting a fileset

Filesets are deleted with the **mmdelfileset** command. There are several notes to keep in mind when deleting filesets:

- The root fileset cannot be deleted.
- A fileset that is not empty cannot be deleted unless the **-f** flag is specified.
- A fileset that is currently linked into the name space cannot be deleted until it is unlinked with the **mmunlinkfileset** command.
- A dependent fileset can be deleted at any time.
- An independent fileset cannot be deleted if it has any dependent filesets or fileset snapshots.
- Deleting a dependent fileset that is included in a fileset or global snapshot removes it from the active file system, but it remains part of the file system in a deleted state.
- Deleting an independent fileset that is included in any global snapshots removes it from the active file system, but it remains part of the file system in a deleted state.
- A fileset in the deleted state is displayed in the **mmlsfileset** output with the fileset name in parenthesis. If the **-L** flag is specified, the latest including snapshot is also displayed. The **--deleted** option of the **mmlsfileset** command can be used to display only deleted filesets.
- The contents of a deleted fileset are still available in the snapshot, through some path name containing a **.snapshots** component, because it was saved when the snapshot was created.
- When the last snapshot that includes the fileset has been deleted, the fileset is fully removed from the file system.

See the **mmdelfileset**, **mmlsfileset**, and **mmunlinkfileset** commands in the *GPFS: Administration and Programming Reference* for complete usage information.

Linking a fileset

After the fileset is created, a junction must be created to link it to the desired location in the file system's namespace using the **mmlinkfileset** command. The file system must be mounted in order to link a fileset. An independent fileset can be linked into only one location anywhere in the namespace, specified by the *JunctionPath* parameter:

- The root directory
- Any subdirectory
- The root fileset or to any other fileset

A dependent fileset can only be linked inside its own inode space.

If *JunctionPath* is not specified, the junction is created in the current directory and has the same name as the fileset being linked. After the command completes, the new junction appears as an ordinary directory, except that the user is not allowed to unlink or delete it with the **rmdir** command it. The user can use the **mv** command on the directory to move to a new location in the parent fileset, but the **mv** command is not allowed to move the junction to a different fileset.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 78

See the **mmlinkfileset** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Unlinking a fileset

A junction to a fileset is removed with the **mmunlinkfileset** command, which unlinks the fileset only from the active directory namespace, but the linked or unlinked state of a fileset in a snapshot is unaffected. The unlink fails if there are files open in the fileset, unless the **-f** option is specified. The root fileset cannot be unlinked.

After issuing the **mmunlinkfileset** command, the fileset can be re-linked to a different parent using the **mmlinkfileset** command. Until the fileset is re-linked, it is not accessible.

Note: If run against a file system that has an unlinked fileset, **mmapplypolicy** will not traverse the unlinked fileset.

Attention: If you are using the TSM Backup-Archive client you must use caution when you unlink filesets that contain data backed up by TSM. TSM tracks files by pathname and does not track filesets. As a result, when you unlink a fileset, it appears to TSM that you deleted the contents of the fileset. Therefore, the TSM Backup-Archive client inactivates the data on the TSM server which may result in the loss of backup data during the expiration process.

See the **mmunlinkfileset** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Changing fileset attributes

To change a fileset's junction, you have to first unlink the fileset using the **mmunlinkfileset** command, and then create the new junction using the **mmlinkfileset** command.

To change the attributes of an existing fileset, including the fileset name, use the **mmchfileset** command.

Note: In an HSM-managed file system, moving or renaming migrated files between filesets will result in recalling of the date from the TSM server.

See the **mmchfileset**, **mmlinkfileset**, and **mmunlinkfileset** commands in the *GPFS: Administration and Programming Reference* for complete usage information.

Displaying fileset information

Fileset status and attributes are displayed with the **mmlsfileset** command. Some of the attributes displayed include:

- Name of the fileset.
- Fileset identifier of the fileset.
- Junction path to the fileset.
- Status of the fileset.
- Root inode number of the fileset.
- Path to the fileset (if linked).
- Inode space.
- User provided comments (if any).

See the **mmlsfileset** command in the *GPFS: Administration and Programming Reference* for complete usage information.

To display the name of the fileset that includes a given file, issue the **mmlsattr** command and specify the **-L** option. See the **mmlsattr** command in the *GPFS: Administration and Programming Reference* for the complete usage information.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 79

Immutability and appendOnly restrictions

To prevent files from being changed or deleted unexpectedly, GPFS provides immutability and appendOnly restrictions.

Applying immutability and appendOnly restrictions to individual files or to directories

You can apply immutability and appendOnly restrictions either to individual files within a fileset or to a directory.

An immutable file cannot be changed or renamed. An appendOnly file allows append operations, but not delete, modify, or rename operations.

An immutable directory cannot be deleted or renamed, and files cannot be added or deleted under such a directory. An appendOnly directory allows new files or subdirectories to be created with 0 byte length; all such new created files and subdirectories are marked as appendOnly automatically.

The **immutable** flag and the **appendOnly** flag can be set independently. If both immutability and appendOnly are set on a file, immutability restrictions will be in effect.

To set or unset these attributes, use the following command options:

mmchattr -i yes|no
Sets or unsets a file to or from an immutable state.

-i yes Sets the **immutable** attribute of the file to **yes**.

-i no Sets the **immutable** attribute of the file to **no**.

mmchattr -a yes|no
Sets or unsets a file to or from an appendOnly state.

-a yes Sets the **appendOnly** attribute of the file to **yes**.

-a no Sets the **appendOnly** attribute of the file to **no**.

Note: Before an immutable or appendOnly file can be deleted, you must change it to mutable or set appendOnly to **no** (by using the **mmchattr** command).

Storage pool assignment of an immutable or appendOnly file can be changed; an immutable or appendOnly file is allowed to transfer from one storage pool to another.

To display whether or not a file is immutable or appendOnly, issue this command:

```
mmlsattr -L myfile
```

The system displays information similar to the following:

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


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 80

The effects of file operations on immutable and appendOnly files

Once a file has been set as immutable or appendOnly, the following file operations and attributes work differently from the way they work on regular files:

delete An immutable or appendOnly file cannot be deleted.

modify/append
An immutable file cannot be modified or appended

An appendOnly file cannot be modified, but it can be appended..

Note: The immutable and appendOnly flag check takes effect after the file is closed; therefore, the file can be modified if it is opened before the file is changed to immutable.

mode An immutable or appendOnly file's mode cannot be changed.

ownership, acl
These attributes cannot be changed for an immutable or appendOnly file.

extended attributes
These attributes cannot be added, deleted, or modified for an immutable or appendOnly file.

timestamp
The timestamp of an immutable or appendOnly file can be changed.

directory
If a directory is marked as immutable, no files can be created, renamed, or deleted under that directory. However, a subdirectory under an immutable directory remains mutable unless it is explicitly changed by **mmchattr**.

If a directory is marked as appendOnly, no files can be renamed or deleted under that directory. However, 0 byte length files can be created.

The following table shows the effects of file operations on an immutable file or an appendOnly file:

Table 4. The effects of file operations on an immutable file or an appendOnly file

Operation	immutable	appendOnly
Add, **delete**, **modify**, or **rename**	No	No
Append	No	Yes
Change ownership, **mode**, or **acl**	No	No
Change atime, **mtime**, or **ctime**	Yes	Yes
Add, **delete**, or **modify extended attributes**	Disallowed by external methods such as **setfattr**. Allowed internally for **dmapi**, **directio**, and others.	Same as for immutable.
Create a file under an immutable or appendOnly directory	No	Yes, 0 byte length only
Rename or **delete** a file under an immutable or appendOnly directory	No	No
Modify a mutable file under an immutable directory	Yes	Not applicable
Set an immutable file back to mutable	Yes	Not applicable
Set an appendOnly file back to a non-appendOnly state	Not applicable	Yes

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 81

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 82

Chapter 3. Creating and maintaining snapshots of GPFS file systems

A snapshot of an entire GPFS file system can be created to preserve the contents of the file system at a single point in time. Snapshots of the entire file system are also known as global snapshots. The storage overhead for maintaining a snapshot is keeping a copy of data blocks that would otherwise be changed or deleted after the time of the snapshot.

Snapshots of a file system are read-only; changes can only be made to the active (that is, normal, non-snapshot) files and directories.

The snapshot function allows a backup or mirror program to run concurrently with user updates and still obtain a consistent copy of the file system as of the time that the snapshot was created. Snapshots also provide an online backup capability that allows easy recovery from common problems such as accidental deletion of a file, and comparison with older versions of a file.

Notes:

1. Because snapshots are not copies of the entire file system, they should not be used as protection against media failures. For information about protection against media failures, see *Recoverability considerations* in the *GPFS: Concepts, Planning, and Installation Guide*.
2. Fileset snapshots provide a method to create a snapshot of an independent fileset instead of the entire file system. For more information about fileset snapshots, see "Fileset-level snapshots" on page 54.
3. A snapshot of a file creates a new file that captures the user data and user attributes from the original. The snapshot file is independent from the original file. For DMAPI managed file systems, the snapshot of a file is not automatically managed by DMAPI, regardless of the state of the original file. The DMAPI attributes from the original file are not inherited by the snapshot. For more information about DMAPI restrictions for GPFS, see the *GPFS: Data Management API Guide*.
4. When snapshots are present, deleting files from the active file system does not always result in any space actually being freed up; rather, blocks may be pushed to the previous snapshot. In this situation, the way to free up space is to delete the oldest snapshot. Before creating new snapshots, it is good practice to ensure that the file system is not close to being full.
5. The use of clones functionally provides writable snapshots. See Chapter 4, "Creating and managing file clones," on page 67.

Management of snapshots of a GPFS file system includes:

- "Creating your GPFS snapshot"
- "Listing GPFS snapshots" on page 62
- "Restoring a GPFS file system from a snapshot" on page 63
- "Using the policy engine to read a snapshot" on page 64
- "Linking to your GPFS snapshots" on page 64
- "Deleting your GPFS snapshot" on page 65

Creating your GPFS snapshot

Use the **mmcrsnapshot** command to create a snapshot of an entire GPFS file system at a single point in time. Snapshots appear in the file system tree as hidden subdirectories of the root.

Global snapshots appear in a subdirectory in the root directory of the file system, whose default name is **.snapshots**. If you prefer to access snapshots from each directory rather than traversing through the root

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 83

directory, you can use an invisible directory to make the connection by issuing the **mmsnapdir** command. See "Linking to your GPFS snapshots" on page 64 for more information.

A snapshot of the file system, *Device*, is identified by a *SnapshotName* name on the **mmcrsnapshot** command. For example, given the file system **fs1** to create a snapshot **snap1**, enter:

```
mmcrsnapshot fs1 snap1
```

The output is similar to this:

```
| Writing dirty data to disk.
| Quiescing all file system operations.
| Writing dirty data to disk again.
| Snapshot snap1 created with id 1.
```

Before issuing the command, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If a second snapshot were to be created at a later time, the first snapshot would remain as is. Snapshots are only made of active file systems, not existing snapshots. For example:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
| Writing dirty data to disk.
| Quiescing all file system operations.
| Writing dirty data to disk again.
| Snapshot snap2 created with id 2.
```

After the command has been issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3

/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3

/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userA/file2
/fs1/.snapshots/snap2/userA/file3
```

See the **mmcrsnapshot** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Listing GPFS snapshots

Use the **mmlssnapshot** command to display existing snapshots of a file system and their attributes.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 84

The **-d** option displays the amount of storage used by a snapshot. GPFS quota management does not take the data blocks used to store snapshots into account when reporting on and determining if quota limits have been exceeded. This is a slow operation and its usage is suggested for problem determination only.

For example, to display the snapshot information for the file system **fs1** with additional storage information, issue this command:

```
mmlssnapshot fs1 -d
```

The system displays information similar to:

```
Snapshots in file system fs1: [data and metadata in KB]
Directory               SnapId    Status          Created                        Data  Metadata
snap1                   1         Valid           Fri Oct 17 10:56:22 2003          0       512
```

See the **mmlssnapshot** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Restoring a GPFS file system from a snapshot

Use the **mmrestorefs** command to restore user data and attribute files in an active file system from the specified global snapshot.

Prior to issuing the **mmrestorefs** command, you must unmount the file system from all nodes in the cluster. The file system may not be remounted until the **mmrestorefs** command has successfully completed, unless you have specified the **-c** option to force the restore to continue even in the event errors are encountered.

Existing snapshots, including the one being used in the restore, are not affected by the **mmrestorefs** command. To obtain a snapshot of the restored file system, you must reissue the **mmcrsnapshot** command.

As an example, let us assume we have a directory structure similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If the directory **userA** is then deleted, we would have:

```
/fs1/file1
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
```

If the directory **userB** is then created using the inode originally assigned to **userA** and we take another snapshot:

```
mmcrsnapshot fs1 snap2
```

The output is similar to this:

```
| Writing dirty data to disk.
| Quiescing all file system operations.
| Writing dirty data to disk again.
| Snapshot snap2 created with id 2.
```

After the command is issued, the directory structure would appear similar to:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 85

```
/fs1/file1
/fs1/userB/file2b
/fs1/userB/file3b
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

If the file system is then restored from **snap1**:

```
mmrestorefs fs1 snap1
```

After the command is issued, the directory structure would appear similar to:

```
/fs1/file1
/fs1/userA/file2
/fs1/userA/file3
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/userA/file2
/fs1/.snapshots/snap1/userA/file3
/fs1/.snapshots/snap2/file1
/fs1/.snapshots/snap2/userB/file2b
/fs1/.snapshots/snap2/userB/file3b
```

See the **mmrestorefs** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Using the policy engine to read a snapshot

You can use the policy engine to read the contents of a snapshot for backup purposes. The **mmapplypolicy** command provides the **-S** option to specify the snapshot during a policy run. Instead of matching rules to the active file system, the policy engine matches the rules against files in the snapshot.

Note: Snapshots are read-only. Policy rules such as MIGRATE or DELETE that make changes or delete files cannot be used with a snapshot.

See the **mmapplypolicy** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Linking to your GPFS snapshots

Snapshot root directories appear in a special **.snapshots** directory under the file system root.

If you prefer to link directly to the snapshot rather than always traverse the root directory, you can use the **mmsnapdir** command with the **-a** option to add a **.snapshots** subdirectory to all directories in the file system. These **.snapshots** subdirectories will contain a link into the corresponding directory for each snapshot that includes the directory in the active file system.

Unlike **.snapshots** in the root directory, however, the **.snapshots** directories added by the **-a** option of the **mmsnapdir** command are invisible in the sense that the **ls** command or **readdir()** function does not return **.snapshots**. This is to prevent recursive file system utilities such as **find** or **tar** from entering into the snapshot tree for each directory they process. For example, if you enter **ls -a /fs1/userA**, the **.snapshots** directory is not listed. However, you can enter **ls /fs1/userA/.snapshots** or **cd /fs1/userA/.snapshots** to confirm that **.snapshots** is present. If a user wants to make one of their snapshot directories more visible, it is suggested to create a symbolic link to **.snapshots**.

The inode numbers that are used for and within these special **.snapshots** directories are constructed dynamically and do not follow the standard rules. These inode numbers are visible to applications

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 86

through standard commands, such as **stat**, **readdir**, or **ls**. The inode numbers reported for these directories can also be reported differently on different operating systems. Applications should not expect consistent numbering for such inodes.

Specifying the **-r** option on the **mmsnapdir** command reverses the effect of the **-a** option, and reverts to the default behavior of a single **.snapshots** directory in the root directory.

The **-s** option allows you to change the name of the **.snapshots** directory. See the **mmsnapdir** command in the *GPFS: Administration and Programming Reference* for complete usage information.

To illustrate this point, assume that a GPFS file system called **fs1**, which is mounted at **/fs1**, has one snapshot called **snap1**. The file system might appear similar to this:

```
/fs1/userA/file2b
/fs1/userA/file3b
/fs1/.snapshots/snap1/userA/file2b
/fs1/.snapshots/snap1/userA/file3b
```

To create links to the snapshots from each directory, and instead of **.snapshots**, use the name **.links**, enter:

```
mmsnapdir fs1 -a -s .links
```

After the command completes, the directory structure would appear similar to:

```
/fs1/userA/file2b
/fs1/userA/file3b
/fs1/userA/.links/snap1/file2b
/fs1/userA/.links/snap1/file3b

/fs1/.links/snap1/userA/file2b
/fs1/.links/snap1/userA/file3b
```

To delete the links, issue:

```
mmsnapdir fs1 -r
```

After the command completes, the directory structure would appear similar to:

```
/fs1/userA/file2b
/fs1/userA/file3b

/fs1/.links/snap1/userA/file2b
/fs1/.links/snap1/userA/file3b
```

See the **mmsnapdir** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Deleting your GPFS snapshot

Use the **mmdelsnapshot** command to delete GPFS snapshots of a file system.

For example, to delete **snap1** for the file system **fs1**, enter:

```
mmdelsnapshot fs1 snap1
```

The output is similar to this:

```
Invalidating snapshot files...
Deleting snapshot files...
 100.00 % complete on Tue Feb 28 10:40:59 2012
Delete snapshot snap1 complete, err = 0
```

See the **mmdelsnapshot** command in the *GPFS: Administration and Programming Reference* for complete usage information.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 87

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 88

Chapter 4. Creating and managing file clones

A file clone is a writable snapshot of an individual file. File clones can be used to provision virtual machines by creating a virtual disk for each machine by cloning a common base image. A related usage is to clone the virtual disk image of an individual machine as part of taking a snapshot of the machine state.

Cloning a file is similar to creating a copy of a file, but the creation process is faster and more space efficient because no additional disk space is consumed until the clone or the original file is modified. Multiple clones of the same file can be created with no additional space overhead. You can also create clones of clones.

Management of file clones in a GPFS file system includes:

- "Creating file clones"
- "Listing file clones" on page 68
- "Deleting file clones" on page 69
- "File clones and disk space management" on page 69
- "File clones and snapshots" on page 69

Creating file clones

File clones can be created from a regular file or a file in a snapshot using the **mmclone** command.

Creating a file clone from a regular file is a two-step process using the **mmclone** command with the **snap** and **copy** keywords:

1. Issue the **mmclone snap** command to create a read-only snapshot of the file to be cloned. This read-only snapshot becomes known as the clone parent. For example, the following command creates a clone parent called **snap1** from the original file **file1**:

   ```
   mmclone snap file1 snap1
   ```

   Alternately, if only one file is specified with the **mmclone snap** command, it will convert the file to a read-only clone parent without creating a separate clone parent file. When using this method to create a clone parent, the specified file cannot be open for writing or have hard links. For example, the following command converts **file1** into a clone parent.

   ```
   mmclone snap file1
   ```

2. Issue the **mmclone copy** command to create a writable clone from a clone parent. For example, the following command creates a writable file clone called **file2** from the clone parent **snap1**:

   ```
   mmclone copy snap1 file2
   ```

Creating a file clone where the source is in a snapshot only requires one step using the **mmclone** command with the **copy** keyword. For example, the following command creates a writable file clone called **file3.clone** from a file called **file3** in a snapshot called **snap2**:

```
mmclone copy /fs1/.snapshots/snap2/file3 file3.clone
```

In this case, **file3** becomes the clone parent.

Note: Extended attributes of clone parents are not passed along to file clones.

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 89

After a clone has been created, the clone and the file that it was cloned from are interchangeable, which is similar to a regular copy (**cp**) command. The file clone will have a new inode number and attributes that can be modified independently of the original file.

Additional clones can be created from the same clone parent by issuing additional **mmclone copy** commands, for example:

```
mmclone copy snap1 file3
```

File clones of clones can also be created, as shown in the following example:

```
mmclone snap file1 snap1
mmclone copy snap1 file2
echo hello >> file2
mmclone snap file2 snap2
mmclone copy snap2 file3
```

The echo command updates the last block of file clone **file2**. When **file2** is snapped to **snap2**, the **mmclone snap** operation is performed as described previously. When a block in **file3** is read, the clone parent inode is found first. For the case of the last block, with the **hello** text, the disk address will be found in **snap2**. However, for other blocks, the disk address will be found in **snap1**.

See the **mmclone** command for complete usage information.

Listing file clones

Use the **mmclone** command to display status for specified files.

The **show** keyword of the **mmclone** command provides a report to determine the current status of one or more files. When a file is a clone, the report will show the parent inode number. When a file was cloned from a file in a snapshot, **mmclone show** displays the snapshot and fileset information.

Consider the following scenario:

1. The **ls** command is issued to show all **img** files in the current directory:

   ```
   ls -ils *.img
   ```

   The system displays output similar to the following:

   ```
   148485 5752576 -rw-r--r-- 1 root root 21474836480 Jan  9 16:19 test01.img
   ```

2. A file clone is then created with the following commands:

   ```
   mmclone snap test01.img base.img
   mmclone copy base.img test02.img
   ```

3. After the file clone is created, the **mmclone show** command is issued to show information about all **img** files in the current directory:

   ```
   mmclone show *.img
   ```

   The system displays output similar to the following:

   ```
   Parent  Depth   Parent inode   File name
   ------  -----  --------------  ---------
      yes      0                  base.img
       no      1          148488  test01.img
       no      1          148488  test02.img
   ```

4. A subsequent **ls** command would display output similar to the following:

   ```
   # ls -ils *.img
   148488 5752576 -rw-r--r-- 3 root root 21474836480 Jan  9 16:25 base.img
   148485       0 -rw-r--r-- 1 root root 21474836480 Jan  9 16:19 test01.img
   148480       0 -rw-r--r-- 1 root root 21474836480 Jan  9 16:25 test02.img
   ```

See the **mmclone** command for complete usage information.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 90

Deleting file clones

There is no explicit GPFS command available for deleting file clones. File clones can be deleted using a regular delete (**rm**) command. Clone parent files cannot be deleted until all file clone copies of the parent have been deleted and all open file handles to them have been closed.

Note: There is a brief period of time, immediately following the deletion of the file clone copies, when deletion of the parent can fail because the clone copy deletions are still running in the background.

Splitting file clones from clone parents

Use the **mmclone** command to split a file clone from a clone parent.

File clones can be split from their clone parents in one of two ways:

- Using the **mmclone redirect** command to split the file clone from the immediate clone parent only. The clone child remains a file clone, but the clone parent can be deleted.
- Using the **mmclone split** command to split the file clone from all clone parents. This converts the former clone child to a regular file. The clone parent does not change.

See the **mmclone** command for complete usage information.

File clones and disk space management

File clones have the following considerations related to disk space management:

- Replication and storage pools

  Each file clone has its own inode, so attributes and permissions for each clone can be set independently. For example, timestamps (atime, mtime, and ctime) are maintained separately for each clone. Clone parent attributes must be changed separately. If different clones have different values for replication or storage pool, it is not possible for every one of the clones to have all data blocks readable through that clone to be replicated and placed consistent with its replication and pool settings. Thus, changes to replication and storage pool only apply to blocks added to the clone and leave the clone parent unchanged.
- Clone ownership, block counts, and quotas

  Creating a clone parent requires read access to the file being cloned. The person creating the clone parent does not have to be the owner of the original file, but will become the owner of the new clone parent. The block count and disk quota of the original file will be transferred to the new clone parent inode and then set to zero in the original file. Any blocks allocated by copy-on-write of a clone file will be added to the block count in the clone inode, with quota charged against the owner of the file. If even a single byte of data in a clone child is changed, the entire block will be copied from the parent.
- Clones and DMAPI

  Clone parent files and clone copy files will be preserved across migrations and recalls to and from tape.

File clones and snapshots

When a snapshot is created and a file clone is subsequently updated, the previous state of the file clone will be saved in the snapshot. When reading a file clone in the snapshot, the system will distinguish between the states of the clone:

- The data block has not been modified since the snapshot was taken, so look for the data in the same file in the next most recent snapshot or active file system.
- The file clone was updated, which indicates that the corresponding data block should be found in the clone parent. The system will look for the data in the clone parent within the same snapshot.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 91

When a snapshot has file clones, those file clones should be deleted or split from their clone parents prior to deleting the snapshot. See “Deleting file clones” on page 69 and “Splitting file clones from clone parents” on page 69 for more information. A policy file can be created to help determine if a snapshot has file clones. See “File clones and policy files” for more information.

File clones and policy files

Policy files can be created to examine clone attributes.

The following clone attributes can be examined in a policy file:

- The depth of the clone tree.
- If file is an immutable clone parent.
- The fileset ID of the clone parent.
- The inode number of the clone parent for the file.
- If the clone parent is in a snapshot.
- The snapshot ID of the clone parent.

See “Using file attributes” on page 26 for more information about the clone attributes available for policy files.

The following example shows a policy file that can be created for displaying clone attributes for all files:

```
RULE EXTERNAL LIST 'x' EXEC ''
RULE 'nonClone' LIST 'x' SHOW('nonclone') WHERE Clone_Parent_Inode IS NULL
RULE 'normalClone' LIST 'x' SHOW(
  'inum ' || varchar(Clone_Parent_Inode) ||
  ' par ' || varchar(Clone_Is_Parent) ||
  ' psn ' || varchar(Clone_Parent_Is_Snap) ||
  ' dep ' || varchar(Clone_Depth))
  WHERE Clone_Parent_Inode IS NOT NULL AND Clone_Parent_Is_Snap == 0
RULE 'snapClone' LIST 'x' SHOW(
  'inum ' || varchar(Clone_Parent_Inode) ||
  ' par ' || varchar(Clone_Is_Parent) ||
  ' psn ' || varchar(Clone_Parent_Is_Snap) ||
  ' dep ' || varchar(Clone_Depth) ||
  ' Fid ' || varchar(Clone_Parent_Fileset_Id) ||
  ' snap ' || varchar(Clone_Parent_Snap_Id))
  WHERE Clone_Parent_Inode IS NOT NULL AND Clone_Parent_Is_Snap != 0
```

If this policy file was called **pol.file**, the following command would display the clone attributes:

```
mmapplypolicy fs0 -P pol.file -I defer -f pol -L 0
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 92

Chapter 5. Scale Out Backup and Restore (SOBAR)

Scale Out Backup and Restore (SOBAR) is a specialized mechanism for data protection against disaster only for GPFS file systems that are managed by Tivoli Storage Manager (TSM) Hierarchical Storage Management (HSM).

Note: This feature is available with GPFS Standard Edition or higher.

To protect a file system against disaster the following steps must be taken to ensure all data is safely stored in a second location:

1. Record the file system configuration with the **mmbackupconfig** command.
2. Ensure all file data is *pre-migrated* (see "Pre-migrating files with external storage pools" on page 49 for more information).
3. Perform a metadata image backup with the **mmimgbackup** command.

The **mmbackupconfig** command must be run prior to running the **mmimgbackup** command. No changes to file system configuration, filesets, quotas, or other settings should be done between running the **mmbackupconfig** command and the **mmimgbackup** command. To recover from a disaster, the **mmrestoreconfig** command must be run prior to running the **mmimgrestore** command. After restoring the image data and adjusting quota settings, the file system can be mounted read-write, and the HSM system re-enabled to permit file data recall. Users may be permitted to access the file system, and/or the system administrator can manually recall file data with the Tivoli HSM command **dsmrecall**.

These commands cannot be run from a Windows node.

Backup procedure with SOBAR

This section provides a detailed example of the backup procedure used with SOBAR.

Throughout these procedures, the sample file system used is called **smallfs**. Where appropriate, replace this value with your file system name.

1. Backup the cluster configuration information.

   The cluster configuration must be backed up by the administrator. The minimum cluster configuration information needed is: IP addresses, node names, roles, quorum and server roles, cluster-wide configuration settings from **mmchconfig**, cluster manager node roles, remote shell configuration, mutual **ssh** and **rsh** authentication setup, and the cluster UID. More complete configuration information can be found in the **mmsdrfs** file.
2. Preserve disk configuration information.

   Disk configuration must also be preserved in order to recover a file system. The basic disk configuration information needed, for a backup intended for disaster recovery, is the number of disk volumes that were previously available and the sizes of those volumes. In order to recover from a complete file system loss, at least as much disk space as was previously available will be needed for restoration. It is only feasible to restore the image of a file system onto replacement disks if the disk volumes available are of similar enough sizes to the originals that all data can be restored to the new disks. At a minimum, the following disk configuration information is needed:
   - Disk device names
   - Disk device sizes
   - The number of disk volumes
   - NSD server configuration
   - Disk RAID configurations

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 93

- Failure group designations
- The **mmsdrfs** file contents

3. Backup the GPFS file system configuration information.

   In addition to the disks, the file system built on those volumes has configuration information that can be captured using the **mmbackupconfig** command. This information includes block size, replication factors, number and size of disks, storage pool layout, filesets and junction points, policy rules, quota information, and a number of other file system attributes. The file system configuration information can be backed up into a single file using a command similar to the following:

   ```
   mmbackupconfig smallfs -o /tmp/smallfs.bkpcfg.out925
   ```

4. Pre-migrate all newer file data into secondary storage.

   File contents in a space-managed GPFS will reside in secondary storage managed by the HSM. In the case of Tivoli Storage Manager HSM, disk and tape pools will typically hold the offline images of migrated files. HSM can also be used to pre-migrate all newer file data into secondary storage, so that all files will have either a migrated or pre-migrated status (**XATTR**) recorded, and their current contents are copied or updated into the secondary storage. The TSM command **dsmmigrate** can be used as follows:

   ```
   dsmmigrate -Premigrate -Recursive /smallfs
   ```

   To optionally check the status of the files that were pre-migrated with the previous command, use the following command:

   ```
   dsmls /smallfs/*
   ```

5. Create a global snapshot of the live file system, to provide a quiescent image for image backup, using a command similar to the following:

   ```
   mmcrsnapshot smallfs smallfssnap
   ```

6. Choose a staging area in which to save the GPFS metadata image files.

   The image backup process stores each piece of the partial file system image backup in its own file in the shared work directory typically used by policy runs. These files can become quite large depending on the number of files in the file system. Also, because the file system holding this shared directory must be accessible to every node participating in the parallel backup task, it might also be a GPFS file system. It is imperative that the staging directory chosen be accessible to both the **tsapolicy** archiver process and the TSM Backup-Archive client. This staging directory is specified with the **-g** option of the **mmimgbackup** command.

7. Backup the file system image.

   The following command will back up an image of the GPFS metadata from the file system using a parallel policy run with the default TSM backup client to backup the file system metadata image:

   ```
   mmimgbackup smallfs -S smallfssnap -g /u/user/backup -N aixnodes
   ```

   The metadata of the file system, the directories, inodes, attributes, symlinks, and so on are all captured in parallel by using the archive module extension feature of the **mmapplypolicy** command. After completing the parallel execution of the policy-driven archiving process, a collection of image files in this format will remain. These image files are gathered by the **mmimgbackup** command and archived to TSM automatically.

   If using the **-N** *nodes* option, it is recommended that the same operating system be used when running **mmimgbackup**. Note the directory created with **-g** *GlobalWorkDirectory* to store the image files.

8. After the image backup is complete, delete the snapshot used for backup with the following command:

   ```
   mmdelsnapshot smallfs smallfssnap
   ```

Restore procedure with SOBAR

This section provides a detailed example of the restore procedure used with SOBAR.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 94

In order to restore a file system, the configuration data stored from a previous run of **mmbackupconfig** and the image files produced from **mmimgbackup** must be accessible.

Throughout these procedures, the sample file system used is called **smallfs**. Where appropriate, replace this value with your file system name.

1. Restore the metadata image files from **mmimgbackup** and the backup configuration data from **mmbackupconfig** with a **dsmc** command similar to the following:

   ```
   dsmc restore -subdir=yes /u/user/backup/8516/
   ```

2. Retrieve the base file system configuration information.

   Use the **mmrestoreconfig** command to generate a configuration file, which contains the details of the former file system:

   ```
   mmrestoreconfig Device -i InputFile -F QueryResultFile
   ```

3. Recreate NSDs if they are missing.

   Using the output file generated in the previous step as a guide, the administrator might need to recreate NSD devices for use with the restored file system. In the output file, the **NSD configuration** section contains the NSD information; for example:

   ```
   ######## NSD configuration ##################
   ## Disk descriptor format for the mmcrnsd command.
   ## Please edit the disk and desired name fields to match
   ## your current hardware settings.
   ##
   ## The user then can uncomment the descriptor lines and
   ## use this file as input to the -F option.
   #
   # %nsd:
   #   device=DiskName
   #   nsd=nsd8
   #   usage=dataAndMetadata
   #   failureGroup=-1
   #   pool=system
   #
   ```

   If changes are needed, edit the file in a text editor and follow the included instructions to use it as input to the **mmcrnsd** command, then issue the following command:

   ```
   mmcrnsd -F StanzaFile
   ```

4. Recreate the base file system.

   The administrator must recreate the initial file system. The output query file specified in the previous commands can be used as a guide. The following example shows the section of this file that is needed when recreating the file system:

   ```
   ######### File system configuration #############
   ## The user can use the predefined options/option values
   ## when recreating the filesystem.  The option values
   ## represent values from the backed up filesystem.
   #
   # mmcrfs FS_NAME NSD_DISKS -j cluster -k posix -Q yes -L 4194304 --disable-fastea
     -T /fs2 -A no --inode-limit 278016#
   #
   # When preparing the file system for image restore, quota
   # enforcement must be disabled at file system creation time.
   # If this is not done, the image restore process will fail.
   ```

   Do one of the following to recreate the file system:

   - Edit the output file. Uncomment the **mmcrfs** command, and specify the appropriate file system name and NSD disk(s). Remove the **-Q** option to ensure quotas are not enabled. Save the changes and run the file as a shell script:

     ```
     sh OutputFile
     ```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 95

- From the command line, issue an **mmcrfs** command similar to the one in the output file, but specify the appropriate file system name and NSD disk(s). Do not specify the **-Q** option to ensure quotas are not enabled.

5. Restore essential file system configuration.

   Using the **mmrestoreconfig** command, the essential file system configuration can be restored to the file system that was just created in the previous step. Quota is disabled in this step because the quota system must remain inactive until after the file system image has been restored. Filesets will also be restored and linked, if necessary, using a method specific for image restore. The **--image-restore** option should be used to restore the configuration data in the proper format for SOBAR; for example:

   ```
   mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925 --image-restore
   ```

6. Mount the file system in read-only mode for image restore with the following command:

   ```
   mmmount smallfs -o ro
   ```

7. Perform the image restore; for example:

   ```
   mmimgrestore smallfs /u/user/backup/8516/mmPolicy.8551.D4D85229
   ```

8. To optionally display the restored file system structure, use the following command:

   ```
   ls -l /smallfs/*
   ```

   The system displays information similar to the following:

   ```
   -rw-r--r-- 1 root root 1024 Sep 25 11:34 /smallfs/1Kfile.1
   -rw-r--r-- 1 root root 1024 Sep 25 11:34 /smallfs/1Kfile.2
   -rwxr--r-- 1 root root  238 Sep 25 11:34 /smallfs/generateChksums*
   ```

9. Unmount the file system with the following command:

   ```
   mmumount smallfs
   ```

10. Restore quota configuration.

    If any quota enforcement was used in the prior file system, it can be restored now using the **mmrestoreconfig** command. This step will not enable quotas if they were not in use at the time of the configuration backup. To restore the quota configuration, issue a command similar to the following:

    ```
    mmrestoreconfig smallfs -i /tmp/smallfs.bkpcfg.out925 -Q only
    ```

11. Mount the file system in read-write mode with the following command:

    ```
    mmmount smallfs
    ```

12. Delete the unusable HSM directory.

    The **.SpaceMan** directory contains file stubs from the former space management control information. This directory must be deleted prior to restarting HSM management. Use the following command:

    ```
    rm -rf /smallfs/.SpaceMan
    ```

13. To optionally restart HSM, use the following command:

    ```
    dsmmigfs restart
    ```

14. Resume HSM management on the newly reconstructed file system, to resume managing disk space and to permit recall of files, with the following command:

    ```
    dsmmigfs add /smallfs
    ```

15. To optionally display the managed file system from HSM, use the following command:

    ```
    dsmls /smallfs/*
    ```

    All files are currently in the migrate state.

16. To optionally begin recalling files by forcing a specific recall, use the following command:

    ```
    dsmrecall -Recursive /smallfs/*
    ```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 96

Chapter 6. Establishing disaster recovery for your GPFS cluster

The ability to detect and quickly recover from a massive hardware failure is of paramount importance to businesses that make use of real-time data processing systems.

GPFS provides a number of features that facilitate the implementation of highly-available GPFS environments capable of withstanding catastrophic hardware failures. By maintaining a replica of the file system's data at a geographically-separate location, the system sustains its processing using the secondary replica of the file system in the event of a total failure in the primary environment.

On a very high level, a disaster-resilient GPFS cluster is made up of two or three, distinct, geographically-separate hardware sites operating in a coordinated fashion. Two of the sites consist of GPFS nodes and storage resources holding a complete replica of the file system. If a third site is active, it consists of a single node and a single disk used as a tiebreaker for GPFS quorum. In the event of a catastrophic hardware failure that disables the operation of an entire site, and assuming the tiebreaker site remains operational, file system services failover to the remaining subset of the cluster and continue serving the data using the replica of the file system that survived the disaster. However if the tiebreaker fails during the disaster, the remaining number of nodes and disks is insufficient to satisfy the quorum rules and the surviving site loses access to the GPFS file system. A manual procedure is needed to instruct GPFS to disregard the existing quorum assignments and continue operating with whatever resources are available.

The secondary replica is maintained by one of several methods:

- Synchronous mirroring utilizing GPFS replication.

  The data and metadata replication features of GPFS are used to implement synchronous mirroring between a pair of geographically-separate sites. The use of logical replication-based mirroring offers a generic solution that relies on no specific support from the disk subsystem beyond the basic ability to read and write data blocks. See *Synchronous mirroring utilizing GPFS replication*.
- Synchronous mirroring utilizing IBM TotalStorage Enterprise Storage Server® (ESS) Peer-to-Peer Remote Copy (PPRC).

  The PPRC feature of the ESS establishes a persistent mirroring relationship between pairs of Logical Units (LUNs) on two subsystems connected over an ESCON® or a Fibre Channel link. All updates performed on the set of primary, or source, LUNs appear in the same order on the secondary, or target, disks in the target subsystem. The PPRC mechanism provides for an exact bitwise replica of the source's content as seen at the time of the failure on the target should the source volume fail. See *Synchronous mirroring utilizing IBM TotalStorage ESS PPRC*.

  Usage of synchronous IBM TotalStorage Enterprise Storage Server (ESS) Peer-to-Peer Remote Copy (PPRC) now extends to IBM TotalStorage Metro Mirror.

  Usage of asynchronous PPRC now extends to IBM TotalStorage Global Mirror.
- Asynchronous mirroring utilizing ESS FlashCopy®.

  Periodic point-in-time copies of the file system are taken using the facilities of ESS FlashCopy. The copy is subsequently transferred to a remote backup location using PPRC, written to tape, or both. See *Asynchronous mirroring utilizing ESS FlashCopy*.

The primary advantage of both synchronous mirroring methods is the minimization of the risk of permanent data loss. Both methods provide two consistent, up-to-date replicas of the file system, each available for recovery should the other one fail. However, inherent to all solutions that synchronously mirror data over a wide area network link is the latency penalty induced by the replicated write I/Os. This makes both synchronous mirroring methods prohibitively inefficient for certain types of

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 97

performance-oriented applications. The asynchronous method effectively eliminates this penalty. However, asynchronous mirroring may result in two distinct and not necessarily consistent replicas of the file system. There are no guarantees as to the validity of data kept in the snapshot beyond the fact that the file system's metadata is consistent and that the user data must have been valid at the time the snapshot was taken.

Synchronous mirroring utilizing GPFS replication

In a configuration utilizing GPFS replication, a single GPFS cluster is defined over three geographically-separate sites consisting of two production sites and a third tiebreaker site. One or more file systems are created, mounted, and accessed concurrently from the two active production sites.

The data and metadata replication features of GPFS are used to maintain a secondary copy of each file system block, relying on the concept of disk failure groups to control the physical placement of the individual copies:

1. Separate the set of available disk volumes into two failure groups. Define one failure group at each of the active production sites.
2. Create a replicated file system. Specify a replication factor of 2 for both data and metadata.

When allocating new file system blocks, GPFS always assigns replicas of the same block to distinct failure groups. This provides a sufficient level of redundancy allowing each site to continue operating independently should the other site fail.

GPFS enforces a node quorum rule to prevent multiple nodes from assuming the role of the file system manager in the event of a network partition. Thus, a majority of quorum nodes must remain active in order for the cluster to sustain normal file system usage. Furthermore, GPFS uses a quorum replication algorithm to maintain the content of the file system descriptor (one of the central elements of the GPFS metadata). When formatting the file system, GPFS assigns some number of disks (usually three) as the descriptor replica holders that are responsible for maintaining an up-to-date copy of the descriptor. Similar to the node quorum requirement, a majority of the replica holder disks must remain available at all times to sustain normal file system operations. This file system descriptor quorum is internally controlled by the GPFS daemon. However, when a disk has failed due to a disaster you must manually inform GPFS that the disk is no longer available and it should be excluded from use.

Considering these quorum constraints, it is suggested that a third site in the configuration fulfill the role of a tiebreaker for the node and the file system descriptor quorum decisions. The tiebreaker site consists of:

1. A single quorum node

   As the function of this node is to serve as a tiebreaker in GPFS quorum decisions, it does not require normal file system access and SAN connectivity. To ignore disk access errors on the tiebreaker node, enable the **unmountOnDiskFail** configuration parameter through the **mmchconfig** command. When enabled, this parameter forces the tiebreaker node to treat the lack of disk connectivity as a local error, resulting in a failure to mount the file system, rather that reporting this condition to the file system manager as a disk failure.
2. A single network shared disk

   The function of this disk is to provide an additional replica of the file system descriptor file needed to sustain quorum should a disaster cripple one of the other descriptor replica disks. Create a network shared disk over the tiebreaker node's internal disk defining:
   - the local node as an NSD server
   - the disk usage as **descOnly**

     The **descOnly** option instructs GPFS to only store file system descriptor information on the disk.

This three-site configuration is resilient to a complete failure of any single hardware site. Should all disk volumes in one of the failure groups become unavailable, GPFS performs a transparent failover to the

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 98

remaining set of disks and continues serving the data to the surviving subset of nodes with no administrative intervention. While nothing prevents you from placing the tiebreaker resources at one of the active sites, to minimize the risk of double-site failures it is suggested you install the tiebreakers at a third, geographically distinct location.

The high-level organization of a replicated GPFS cluster for synchronous mirroring where all disks are directly attached to all nodes in the cluster is shown in Figure 5. An alternative to this design would be to have the data served through designated NSD servers.

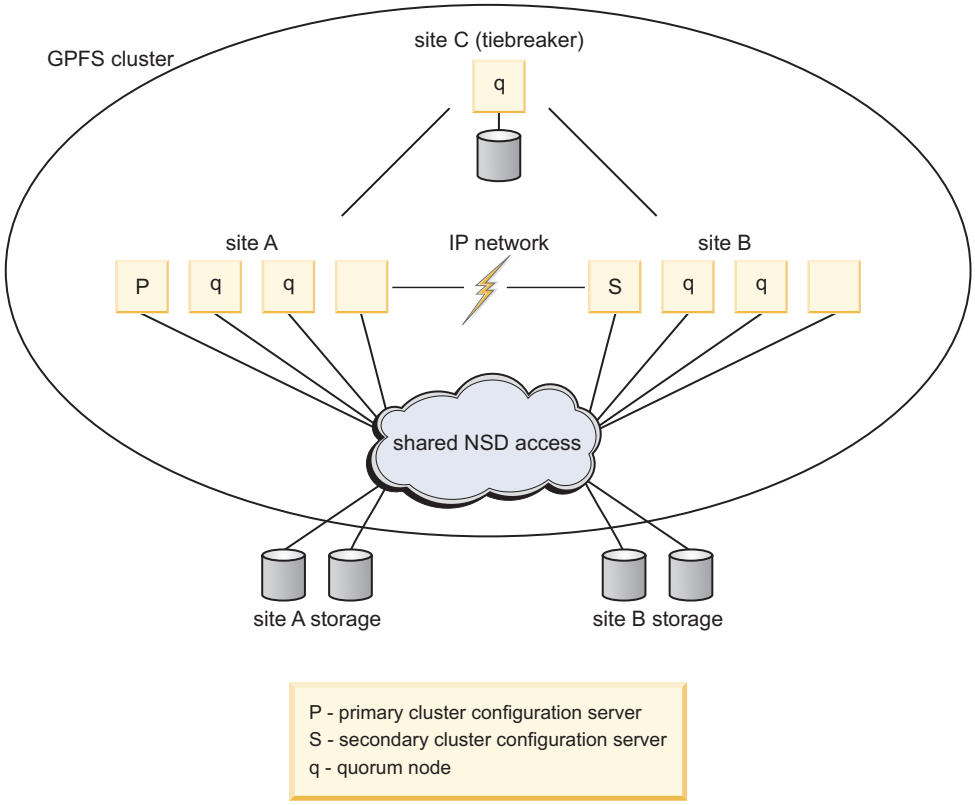


Figure 5. Synchronous mirroring utilizing GPFS replication

Setting up a GPFS cluster with synchronous mirroring utilizing GPFS replication

To establish a disaster-resilient GPFS cluster utilizing replication as shown in Figure 5, consider the configuration:

Site A Consisting of:

- Nodes – **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**
- Disk device names – **diskA1**, **diskA2**

  diskA1 and **diskA2** are SAN-attached and accessible from all nodes at site **A** and site **B**.

Site B Consisting of:

- Nodes – **nodeB001**, **nodeB002**, **nodeB003**, **nodeB004**
- Disks – **diskB1**, **diskB2**

  diskB1 and **diskB2** are SAN-attached and accessible from all nodes at site **A** and site **B**.

Site C (tiebreaker)

Consisting of:

- Node – **nodeC**

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 99

- Disk – **diskC**

  diskC is an NSD defined over the internal disk of the node **nodeC** and is directly accessible only from site **C**

1. Create a GPFS cluster selecting **nodeA001** at site A as the primary cluster data server node, **nodeB001** at site **B** as the secondary cluster data server nodes, and the nodes in the cluster contained in the file **clusterNodes**. The **clusterNodes** file contains the node descriptors:

   ```
   nodeA001:quorum-manager
   nodeA002:quorum-manager
   nodeA003:quorum-manager
   nodeA004:client
   nodeB001:quorum-manager
   nodeB002:quorum-manager
   nodeB003:quorum-manager
   nodeB004:client
   nodeC:quorum-client
   ```

   Issue this command:

   ```
   mmcrcluster –N clusterNodes –p nodeA001 –s nodeB001
   ```

2. Prevent false disk errors in the SAN configuration from being reported to the file system manager by enabling the **unmountOnDiskFail** option on the tiebreaker node:

   ```
   mmchconfig unmountOnDiskFail=yes -N nodeC
   ```

3. Define the set of network shared disks for the cluster where disks at sites **A** and **B** are assigned to failure groups 1 and 2, respectively. The tiebreaker disk is assigned to failure group 3. The disk descriptors contained in the file **clusterDisks** are:

   ```
   /dev/diskA1:nodeA002:nodeA003:dataAndMetadata:1
   /dev/diskA2:nodeA002:nodeA003:dataAndMetadata:1
   /dev/diskB1:nodeB002:nodeB003:dataAndMetadata:2
   /dev/diskB2:nodeB002:nodeB003:dataAndMetadata:2
   /dev/diskC1:nodeC::descOnly:3
   ```

   Issue this command:

   ```
   mmcrnsd –F clusterDisks
   ```

4. Issue the **mmlsnsd** command to verify that the network shared disks have been created:

   ```
   mmlsnsd -m
   ```

   Output is similar to this:

   ```
   Disk name    NSD volume ID      Device         Node name         Remarks
   ---------------------------------------------------------------------------------
    gpfs1nsd     0972445B416BE502   /dev/diskA1    nodeA002          server node
    gpfs1nsd     0972445B416BE502   /dev/diskA1    nodeA003          server node
    gpfs2nsd     0972445B416BE509   /dev/diskA2    nodeA002          server node
    gpfs2nsd     0972445B416BE509   /dev/diskA2    nodeA003          server node
    gpfs3nsd     0972445F416BE4F8   /dev/diskB1    nodeB002          server node
    gpfs3nsd     0972445F416BE4F8   /dev/diskB1    nodeB003          server node
    gpfs4nsd     0972445F416BE4FE   /dev/diskB2    nodeB002          server node
    gpfs4nsd     0972445F416BE4FE   /dev/diskB2    nodeB003          server node
    gpfs5nsd     0972445D416BE504   /dev/diskC1    nodeC             server node
   ```

5. Start the GPFS daemon on all nodes:

   ```
   mmstartup -a
   ```

6. Create a replicated file system **fs0**:

   ```
   mmcrfs /gpfs/fs0 fs0 –F clusterDisks –m 2 –M 2 –r 2 –R 2
   ```

7. Mount **fs0** on all nodes at sites A and B.

Steps to take after a disaster when using GPFS replication

Utilizing GPFS replication allows for *failover* to the surviving site without disruption of service as long as both the remaining site and the tiebreaker site remain functional. It remains in this state until a decision

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 100

is made to restore the operation of the affected site by executing the *failback* procedure. If the tiebreaker site is also affected by the disaster and is no longer operational, GPFS quorum is broken and manual intervention is required to resume file system access. Existing quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements:

1. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to non-quorum nodes. Issue the **mmchnode --nonquorum** command.
2. To relax file system descriptor quorum, temporarily eliminate the failed disks from the group of disks from which the GPFS daemon uses to write the file system descriptor file to. Issue the **mmfsctl exclude** command for each of the failed disks.

While the GPFS cluster is in a failover state, it is suggested that no changes to the GPFS configuration be made. Changes to your GPFS configuration require both cluster configuration servers to be operational. If both servers are not operational, the sites would have distinct, and possibly inconsistent, copies of the GPFS **mmsdrfs** configuration data file. While the servers can be migrated to the surviving site, it is best to avoid this step if the disaster does not leave the affected site permanently disabled.

If it becomes absolutely necessary to modify the GPFS configuration while in failover mode, for example to relax quorum, you must ensure that all nodes at the affected site are powered down and left in a stable inactive state. They must remain in such state until the decision is made to execute the failback procedure. As a means of precaution, we suggest disabling the GPFS autoload option on all nodes to prevent GPFS from bringing itself up automatically on the affected nodes should they come up spontaneously at some point after a disaster.

Failover to the surviving site

Following a disaster, which failover process is implemented depends upon whether or not the tiebreaker site is affected:

- **Failover without the loss of tiebreaker site C**

  The proposed three-site configuration is resilient to a complete failure of any single hardware site. Should all disk volumes in one of the failure groups become unavailable, GPFS performs a transparent failover to the remaining set of disks and continues serving the data to the surviving subset of nodes with no administrative intervention.

- **Failover with the loss of tiebreaker site C**

  If both site **A** and site **C** fail:

  1. Shut the GPFS daemon down on the surviving nodes at site **B**, where the file **gpfs.siteB** lists all of the nodes at site **B**:

     ```
     mmshutdown -N gpfs.siteB
     ```

  2. If it is necessary to make changes to the configuration, migrate the primary cluster configuration server to a node at site **B**:

     ```
     mmchcluster -p nodeB002
     ```

  3. Relax node quorum by temporarily changing the designation of each of the failed quorum nodes to non-quorum nodes:

     ```
     mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC
     ```

  4. Relax file system descriptor quorum by informing the GPFS daemon to migrate the file system descriptor off of the failed disks:

     ```
     mmfsctl fs0 exclude -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
     ```

  5. Restart the GPFS daemon on the surviving nodes:

     ```
     mmstartup -N gpfs.siteB
     ```

  6. Mount the file system on the surviving nodes at site B.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 101

Failback procedures

Which failback procedure you follow depends upon whether the nodes and disks at the affected site have been repaired or replaced. If the disks have been repaired, you must also consider the state of the data on the failed disks:

- For nodes and disks that have been repaired and *you are certain* the data on the failed disks has not been changed, follow either:
  - *failback with temporary loss and no configuration changes*
  - *failback with temporary loss and configuration changes*
- If the nodes have been replaced and either the disks have been replaced or repaired, and *you are not certain* the data on the fail disks has not been changed, follow the procedure for *failback with permanent loss*.

Delayed failures: In certain failure cases the loss of data may not be immediately apparent. For example, consider this sequence of events:

1. Site **B** loses connectivity with sites **A** and **C**.
2. Site **B** then goes down due to loss of node quorum.
3. Sites **A** and **C** remain operational long enough to modify some of the data on disk but suffer a disastrous failure shortly afterwards.
4. Node and file system descriptor quorums are overridden to enable access at site **B**.

Now the two replicas of the file system are inconsistent and the only way to reconcile these copies during recovery is to:

1. Remove the damaged disks at sites **A** and **C**.
2. Either replace the disk and format a new NSD or simply reformat the existing disk if possible.
3. Add the disk back to the file system, performing a full resynchronization of the file system's data and metadata and restore the replica balance using the **mmrestripefs** command.

Failback with temporary loss and no configuration changes:
If the outage was of a temporary nature and your configuration has not been altered, it is a simple process to failback to the original state. After all affected nodes and disks have been repaired and *you are certain* the data on the failed disks has not been changed:

1. Start GPFS on the repaired nodes where the file **gpfs.sitesAC** lists all of the nodes at sites **A** and **C**:
   ```
   mmstartup -N gpfs.sitesAC
   ```
2. Restart the affected disks. If more than one disk in the file system is down, they must all be started at the same time:
   ```
   mmchdisk fs0 start -a
   ```

Failback with temporary loss and configuration changes:
If the outage was of a temporary nature and your configuration has been altered, follow this procedure to failback to the original state. After all affected nodes and disks have been repaired and *you are certain* the data on the failed disks has not been changed:

1. Ensure that all nodes have the latest copy of the **mmsdrfs** file:
   ```
   mmchcluster -p LATEST
   ```
2. Migrate the primary cluster configuration server back to site **A**:
   ```
   mmchcluster -p nodeA001
   ```
3. Restore node quorum designations at sites **A** and **C**:
   ```
   mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
   ```
4. Start GPFS on the repaired nodes where the file **gpfs.sitesAC** lists all of the nodes at sites **A** and **C**:
   ```
   mmstartup -N gpfs.sitesAC
   ```
5. Restore the file system descriptor quorum by informing the GPFS to include the repaired disks:
   ```
   mmfsctl fs0 include -d "gpfs1nsd;gpfs2nsd;gpfs5nsd"
   ```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 102

6. Bring the disks online and restripe the file system across all disks in the cluster to restore the initial replication properties:

```
mmchdisk fs0 start -a
mmrestripefs fs0 -b
```

   The **-r** flag may be used on the **mmrestripefs** command instead.

Failback with permanent loss:
If the outage is of a permanent nature:

1. Remove the failed resources from the GPFS configuration
2. Replace the failed resources, then add the new resources into the configuration
3. Resume the operation of GPFS across the entire cluster

Assume that sites **A** and **C** have had permanent losses. To remove all references of the failed nodes and disks from the GPFS configuration and replace them :

1. To remove the failed resources from the GPFS configuration:
   a. If as part of the failover process, *you did not* migrate the primary cluster configuration server, migrate the server to node **nodeB002** at site B:

      ```
      mmchcluster –p nodeB002
      ```

   b. Delete the failed disks from the GPFS configuration:

      ```
      mmdeldisk fs0 "gpfs1nsd;gpfs2nsd;gpfs5nsd"
      mmdelnsd "gpfs1nsd;gpfs2nsd;gpfs5nsd"
      ```

   c. Delete the failed nodes from the GPFS configuration:

      ```
      mmdelnode -N nodeA001,nodeA002,nodeA003,nodeA004,nodeC
      ```

2. If there are new resources to add to the configuration:
   a. Add the new nodes at sites **A** and **C** to the cluster where the file **gpfs.sitesAC** lists of the new nodes:

      ```
      mmaddnode -N gpfs.sitesAC
      ```

   b. Ensure that all nodes have the latest copy of the **mmsdrfs** file:

      ```
      mmchcluster -p LATEST
      ```

   c. Migrate the primary cluster configuration server back to site **A**:

      ```
      mmchcluster -p nodeA001
      ```

   d. Start GPFS on the new nodes

      ```
      mmstartup -N gpfs.sitesAC
      ```

   e. Prepare the new disks for use in the cluster, create the NSDs using the original disk descriptors for site **A** contained in the file **clusterDisksAC**:

      ```
      /dev/diskA1:nodeA002:nodeA003:dataAndMetadata:1
      /dev/diskA2:nodeA002:nodeA003:dataAndMetadata:1
      /dev/diskC1:nodeC::descOnly:3
      ```

      Issue this command:

      ```
      mmcrnsd -F clusterDisksAC
      ```

   f. Add the new NSDs to the file system specifying the **-r** option to rebalance the data on all disks:

      ```
      mmadddisk fs0 -F clusterDisksAC -r
      ```

Synchronous mirroring utilizing IBM TotalStorage ESS PPRC

PPRC is a function that continuously updates a secondary (target) copy of an ESS disk volume to match changes made to a primary (source) volume. Any pair of equal-sized ESS disks can be configured for a PPRC relationship, during which all write operations performed on the source are synchronously mirrored to the target device.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 103

The PPRC protocol guarantees that the secondary copy is constantly up-to-date by ensuring that the primary copy is written only if the primary storage subsystem received acknowledgement that the secondary copy has been written. The paired volumes typically reside on two distinct and geographically separated ESS devices communicating over ESCON or over a Fibre Channel link.

A number of PPRC tasks are provided to facilitate recovery in the event of a site-wide failure. After the failure of the primary volume (or the failure of the entire storage subsystem), users execute the PPRC failover task, which suspends the PPRC relationship between the given pair of volumes and turns the target volume into a primary. When a volume enters the suspended state, a modification bitmap is established to keep track of the write operations performed on that volume to allow for an efficient resynchronization.

Once the operation of the original primary volume has been restored, the PPRC failback task is executed to resynchronize the content of the two volumes. The original source volume is switched to the target mode, after which all modified data tracks (those recorded in the modification bitmap) are copied from the original target disk. The volume pair is then suspended again and another task is executed to reverse the volumes' roles, thus bringing the pair into its initial state.

The ESS Copy Services Web Interface User's Guide as described in the *IBM TotalStorage Enterprise Storage Server Web Interface User's Guide*, is a GUI that allows users to establish and terminate PPRC pairs, and invoke failover, failback, and related PPRC functions. A Java-based command-line interface as described in the *IBM Enterprise Storage Server Command-Line Interfaces User's Guide*, provides another method of interaction.

A PPRC-based GPFS cluster can be established in two manners:

- A single GPFS cluster encompassing two sites and an optional tiebreaker site
- Two distinct GPFS clusters

An active/active GPFS cluster

The high-level organization of PPRC-based active/active GPFS cluster is illustrated in Figure 6 on page 83. A single GPFS cluster is created over three sites. The data is mirrored between two active sites with a cluster configuration server residing at each site and a tiebreaker quorum node installed at the third location. The presence of an optional tiebreaker node allows the surviving site to satisfy the node quorum requirement with no additional intervention. Without the tiebreaker, the failover procedure requires an additional administrative command to relax node quorum and allow the remaining site to function independently. Furthermore, the nodes at the recovery site have direct disk paths to the primary site's storage.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 104

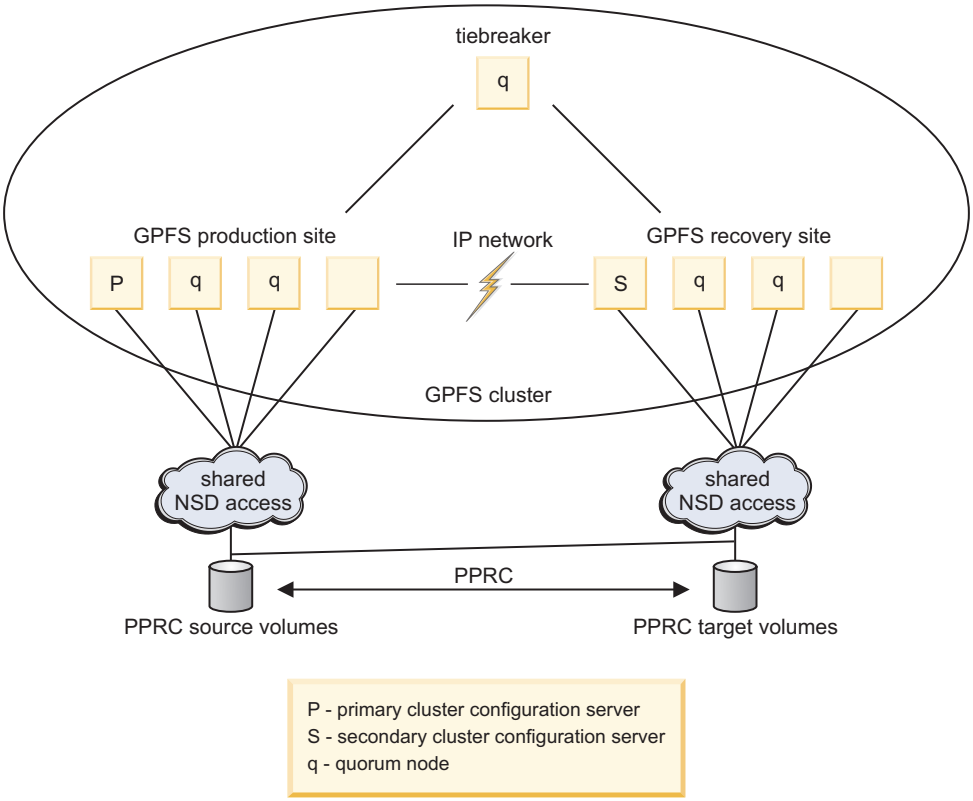


Figure 6. A synchronous active/active PPRC-based mirrored GPFS configuration with a tiebreaker site

Setting up an active/active GPFS configuration

To establish an active/active PPRC-based GPFS cluster with a tiebreaker site as shown in Figure 6, consider the configuration:

Site A (production site)

Consists of:

- Nodes – **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**
- Storage subsystems – Enterprise Storage Server (ESS) **A**, logical subsystem (LSS) **A**
- Disk volumes – **diskA** on LSS **A**

  diskA is SAN-attached and accessible from sites **A** and **B**

Site B (recovery site)

Consists of:

- Nodes – **nodeB001**, **nodeB002**, **nodeB003**, **nodeB004**
- Storage subsystems – ESS **B**, LSS **B**
- Disk volumes – **diskB** on LSS **B**

  diskB is SAN-attached and accessible from site **B** only

Site C (tiebreaker)

Consists of:

- Nodes – **nodeC**

  diskC is an NSD defined over the internal disk of the node **nodeC** and is directly accessible only from site **C**

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 105

1. Create a dual-active ESS copy services domain assigning ESS A as **ServerA** and ESS B as **ServerB**. See the *IBM Enterprise Storage Server User's Guide*. In order to provide for the availability of at least one server after a disaster, the servers should reside at different sites.
2. Establish a PPRC logical path from LSS **A** to LSS **B**. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
3. In order to protect the order of dependent writes that span multiple disk volumes, multiple LSS devices, or both, a consistency group should be defined over all logical subsystems at the primary site. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*. In this case that would only be LSS A. See *Data integrity and the use of PPRC consistency groups*.
4. Establish a synchronous PPRC volume pair between the source and target using the **copy entire volume** option and leave the **permit read from secondary** option disabled. In this case it would be diskA-diskB. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
5. Create a GPFS cluster defining the primary cluster configuration server as nodes **nodeA001** at site **A**, the secondary cluster configuration server as **nodeB001** at site **B**, an equal number of quorum nodes at each site, including the tiebreaker node at site **C**, **nodeC**. To prevent the tiebreaker node from assuming the role of file system manager, define it as **client**. Define all other quorum nodes as **manager**. List the nodes in the cluster in the file **NodeDescFile**. The **NodeDescFile** file contains the node descriptors:

   ```
   nodeA001:quourum-manager
   nodeA002:quorum-manager
   nodeA003:quorum-manager
   nodeA004:client
   nodeB001:quorum-manager
   nodeB002:quorum-manager
   nodeB003:quorum-manager
   nodeB004:client
   nodeC:quorum-client
   ```

   Issue this command:

   ```
   mmcrcluster –N NodeDescFile –p nodeA001 –s nodeB001
   ```
6. Enable the **unmountOnDiskFail** option on the tiebreaker node preventing false disk errors in the SAN configuration from being reported to the file system manager by issuing the **mmchconfig** command:

   ```
   mmchconfig unmountOnDiskFail=yes -N nodeC
   ```
7. Create an NSD over **diskA**. The disk descriptor contained in the file **DiskDescFile** is:

   ```
   /dev/diskA:nodeA001:nodeA002:dataAndMetadata:1
   ```

   Issue this command:

   ```
   mmcrnsd –F DiskDescFileP
   ```
8. Start the GPFS daemon on all nodes:

   ```
   mmstartup -a
   ```
9. Create a GPFS file system and mount it on all nodes at sites **A** and **B**.

   ```
   mmcrfs /gpfs/fs0 fs0 -F DiskDescFile
   ```

Failover to the recovery site and subsequent failback for an active/active configuration

For an active/active PPRC-based cluster, follow these steps to restore access to the file system through site **B** after site **A** has experienced a disastrous failure:

1. Stop the GPFS daemon on the surviving nodes as site **B** where the file **gpfs.siteB** lists all of the nodes at site **B**:

   ```
   mmshutdown -N gpfs.siteB
   ```
2. Perform PPRC failover, establishing a synchronous PPRC pair **diskB-diskA** with the PPRC failover option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*. The

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 106

PPRC state of **diskB** is changed from *duplex target* to *suspended source* and is available for regular I/O. Refer to the *IBM Enterprise Storage Server* for a detailed explanation of PPRC failover.

3. If you needed to relax node quorum or make configuration changes, migrate the primary cluster configuration server to site **B**, issue this command:

   ```
   mmchcluster -p nodeB001
   ```

4. If site **C**, the tiebreaker, failed along with site **A**, existing node quorum designations must be relaxed in order to allow the surviving site to fulfill quorum requirements. To relax node quorum, temporarily change the designation of each of the failed quorum nodes to non-quorum nodes:

   ```
   mmchnode --nonquorum -N nodeA001,nodeA002,nodeA003,nodeC
   ```

5. Ensure the source volumes are *not* accessible to the recovery site:
   - Disconnect the cable
   - Define the **nsddevices** user exit file to exclude the source volumes
6. Restart the GPFS daemon on all surviving nodes:

   ```
   mmstartup -N gpfs.siteB
   ```

Once the operation of site **A** has been restored, the failback procedure is executed to restore the access to the file system from that location. The failback operation is a two-step process:

1. Resynchronize the paired volumes by establishing a temporary PPRC pair with **diskB** defined as the source and **diskA** as the target. The modification bitmap is traversed to find the mismatching disk sectors, whose content is then copied from **diskB** to **diskA**.
   a. Establish a PPRC path from LSS **B** to LSS **A** enabling the **consistency group** option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   b. Establish a synchronous PPRC pair **diskB-diskA** with the PPRC **failback** option and **permit read from secondary** left disabled. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   c. Wait for the volume pair to reach the *duplex* (fully synchronized) state.
2. Shut GPFS down at site **B** and reverse the disk roles (the original primary disk becomes the primary again), bringing the PPRC pair to its initial state.
   a. Stop the GPFS daemon on all nodes.
   b. Establish a synchronous PPRC pair **diskA-diskB** with the PPRC failover option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   c. Establish a synchronous PPRC pair **diskA-diskB** with the PPRC failback option and **permit read from secondary** left disabled. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   d. If during failover you migrated the primary cluster configuration server to a node in site **B**:
      1) Migrate the primary cluster configuration server back to site **A**:

         ```
         mmchcluster -p nodeA001
         ```

      2) Restore the initial quorum assignments:

         ```
         mmchnode --quorum -N nodeA001,nodeA002,nodeA003,nodeC
         ```

      3) Ensure that all nodes have the latest copy of the **mmfsdr** file:

         ```
         mmchcluster -p LATEST
         ```

   e. Ensure the source volumes *are* accessible to the recovery site:
      - Reconnect the cable
      - Edit the **nsddevices** user exit file to *include* the source volumes
   f. Start the GPFS daemon on all nodes:

      ```
      mmstartup -a
      ```

   g. Mount the file system on all the nodes at sites **A** and **B**.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 107

An active/passive GPFS cluster

In an active/passive environment, two GPFS clusters are set up in two geographically distinct locations (the production and the recovery sites). We refer to these as peer GPFS clusters. A GPFS file system is defined over a set of disk volumes located at the production site and these disks are mirrored using PPRC to a secondary set of volumes located at the recovery site. During normal operation, only the nodes in the production GPFS cluster mount and access the GPFS file system at any given time, which is the primary difference between a configuration of this type and the active/active model.

In the event of a catastrophe in the production cluster, the PPRC failover task is executed to enable access to the secondary replica of the file system located on the target PPRC disks. See IBM Redbooks® for *IBM TotalStorage Enterprise Storage Server Implementing ESS Copy Services in Open Environments* for a detailed explanation of PPRC failover and failback and the means of invoking these procedures in your environment.

The secondary replica is then mounted on nodes in the recovery cluster as a regular GPFS file system, thus allowing the processing of data to resume at the recovery site. At a latter point, after restoring the physical operation of the production site, we execute the failback procedure to resynchronize the content of the PPRC volume pairs between the two clusters and re-enable access to the file system in the production environment.

The high-level organization of synchronous active/passive PPRC-based GPFS cluster is shown in Figure 7.

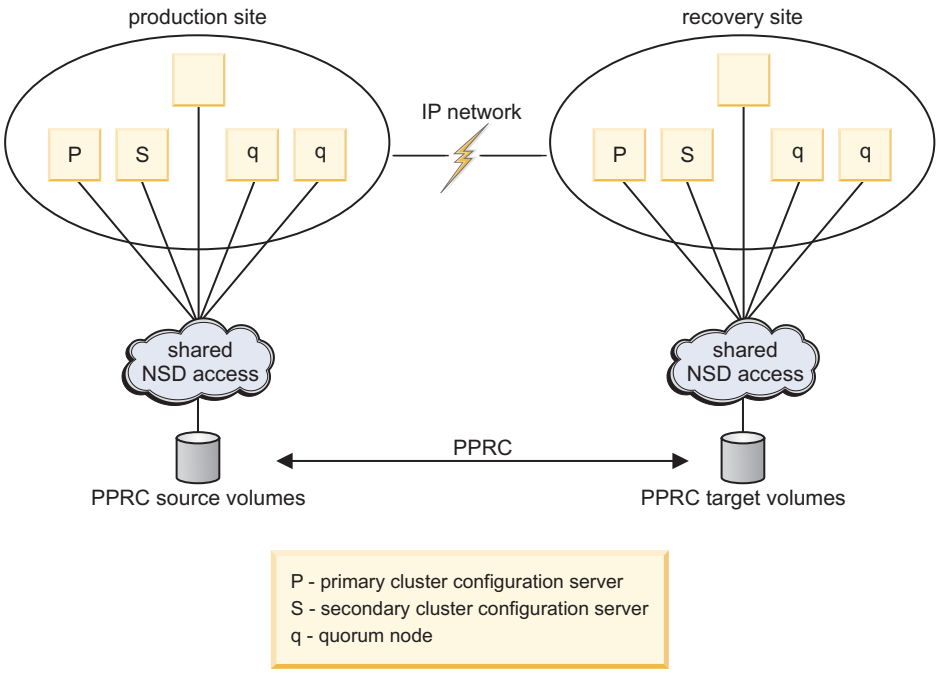


Figure 7. A synchronous active/passive PPRC-based GPFS configuration without a tiebreaker site

Setting up an active/passive GPFS configuration

To establish an active/passive PPRC based GPFS cluster as shown in Figure 7, consider the configuration:

Production site

Consists of:

- Nodes – **nodeP001**, **nodeP002**, **nodeP003**, **nodeP004**, **nodeP005**
- Storage subsystems – Enterprise Storage Server (ESS) **P**, logical subsystem (LSS) **P**

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 108

- LUN ids and disk volume names – **lunP1** (**hdisk11**), **lunP2** (**hdisk12**), **lunP3** (**hdisk13**), **lunP4** (**hdisk14**)

Recovery site

Consists of:

- Nodes – **nodeR001**, **nodeR002**, **nodeR003**, **nodeR004**, **nodeR005**
- Storage subsystems – ESS **R**, LSS **R**
- LUN ids and disk volume names – **lunR1** (**hdisk11**), **lunR2** (**hdisk12**), **lunR3** (**hdisk13**), **lunR4** (**hdisk14**)

All disks are SAN-attached and directly accessible from all local nodes.

1. Create a dual-active ESS copy services domain assigning ESS **P** as **ServerA** and ESS **R** as **ServerB**. See the *IBM Enterprise Storage Server User's Guide*.
2. Establish a PPRC logical path from LSS **P** to LSS **R** enabling the consistency group option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*, section on *Data integrity and the use of PPRC consistency groups*.
3. Establish synchronous PPRC volume pairs using the **copy entire volume** option and leave the **permit read from secondary** option disabled:

   ```
   lunP1-lunR1 (source-target)
   lunP2-lunR2 (source-target)
   lunP3-lunR3 (source-target)
   lunP4-lunR4 (source-target)
   ```

   See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
4. Create the recovery cluster selecting **nodeR001** as the primary cluster data server node, **nodeR002** as the secondary cluster data server nodes, and the nodes in the cluster contained in the file **NodeDescFileR**. The **NodeDescFileR** file contains the node descriptors:

   ```
   nodeR001:quourum-manager
   nodeR002:quourum-manager
   nodeR003:quourum-manager
   nodeR004:quourum-manager
   nodeR005
   ```

   Issue this command:

   ```
   mmcrcluster –N NodeDescFileR –p nodeR001 –s nodeR002
   ```
5. Create the GPFS production cluster selecting **nodeP001** as the primary cluster data server node, **nodeP002** as the secondary cluster data server node, and the nodes in the cluster contained in the file **NodeDescFileP**. The **NodeDescFileP** file contains the node descriptors:

   ```
   nodeP001:quourum-manager
   nodeP002:quourum-manager
   nodeP003:quourum-manager
   nodeP004:quourum-manager
   nodeP005
   ```

   Issue this command:

   ```
   mmcrcluster –N NodeDescFileP –p nodeP001 –s nodeP002
   ```
6. At all times the peer clusters must see a consistent image of the mirrored file system's configuration state contained in the **mmsdrfs** file. After the initial creation of the file system, all subsequent updates to the local configuration data must be propagated and imported into the peer cluster. Execute the **mmfsctl syncFSconfig** command to resynchronize the configuration state between the peer clusters after each of these actions in the primary GPFS cluster:
   - Addition of disks through the **mmadddisk** command
   - Removal of disks through the **mmdeldisk** command
   - Replacement of disks through the **mmrpldisk** command

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 109

- Modifications to disk attributes through the **mmchdisk** command
- Changes to the file system's mount point through the **mmchfs -T** command

To automate the propagation of the configuration state to the recovery cluster, activate and use the **syncFSconfig** user exit. Follow the instructions in the prolog of **/usr/lpp/mmfs/samples/syncfsconfig.sample**.

7. From a node in the production cluster, start the GPFS daemon on all nodes:

   ```
   mmstartup -a
   ```

8. Create the NSDs at the production site. The disk descriptors contained in the file **DiskDescFileP** are:

   ```
   /dev/hdisk11:nodeP001:nodeP002:dataAndMetadata:-1
   /dev/hdisk12:nodeP001:nodeP002:dataAndMetadata:-1
   /dev/hdisk13:nodeP001:nodeP002:dataAndMetadata:-1
   /dev/hdisk14:nodeP001:nodeP002:dataAndMetadata:-1
   ```

   Issue this command:

   ```
   mmcrnsd –F DiskDescFileP
   ```

9. Create the GPFS file system and mount it on all nodes at the production site:

   ```
   mmcrfs /gpfs/fs0 fs0 -F DiskDescFileP
   ```

Failover to the recovery site and subsequent failback for an active/passive configuration

For an active/passive PPRC-based cluster, follow these steps to failover production to the recovery site:

1. If the GPFS daemon is not already stopped on all surviving nodes in the production cluster, from a node in the production cluster issue:

   ```
   mmshutdown –a
   ```

2. Perform PPRC failover. This step involves establishing synchronous PPRC pairs with the failover option:
   - **lunR1-lunP1**
   - **lunR2-lunP2**
   - **lunR3-lunP3**
   - **lunR4-lunP4**

   See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*. After the completion of this step, the volumes on ESS R enter the **suspended source** PPRC state.

3. From a node in the recovery cluster start GPFS:

   ```
   mmstartup –a
   ```

4. Mount the file system on all nodes in the recovery cluster.

Once the physical operation of the production site has been restored, execute the failback procedure to transfer the file system activity back to the production GPFS cluster. The failback operation is a two-step process:

1. For each of the paired volumes, resynchronize the pairs by establishing a temporary PPRC pair with the recovery LUN acting as the sources for the production LUN. The PPRC modification bitmap is traversed to find the mismatching disk tracks, whose content is then copied from the recovery LUN to the production LUN.
   a. Establish a PPRC path from LSS **R** to LSS **P** enabling the **consistency group** option. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
   b. Establish synchronous PPRC volume pairs with the **failback** option, leaving the **permit read from secondary** option disabled.
      - **lunR1-lunP1**
      - **lunR2-lunP2**
      - **lunR3-lunP3**

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 110

- **lunR4-lunP4**

See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.

c. Wait for all PPRC pairs to reach the **duplex** (fully synchronized) state.

d. If the state of the system configuration has changed, update the GPFS configuration data in the production cluster to propagate the changes made while in failover mode. From a node at the recovery site, issue:

```
mmfsctl all syncFSconfig –n gpfs.sitePnodes
```

2. Stop GPFS on all nodes in the recovery cluster and reverse the disk roles so the original primary disks become the primaries again:

a. From a node in the recovery cluster, stop the GPFS daemon on all nodes in the recovery cluster:

```
mmshutdown –a
```

b. Establish synchronous PPRC volume pairs with the PPRC failover option:

- **lunP1-lunR1**
- **lunP2-lunR2**
- **lunP3-lunR3**
- **lunP4-lunR4**

See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.

c. Establish synchronous PPRC volume pairs with the PPRC failback option, leaving the permit **read from secondary** option disabled:

- **lunP1-lunR1**
- **lunP2-lunR2**
- **lunP3-lunR3**
- **lunP4-lunR4**

See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.

d. From a node in the production cluster, start GPFS:

```
mmstartup –a
```

e. From a node in the production cluster, mount the file system on all nodes in the production cluster.

Data integrity and the use of PPRC consistency groups

The integrity of the post-disaster replica of the file system contained on the secondary PPRC disk volumes depends on the assumption that the order of dependent write operations is preserved by the PPRC mirroring mechanism. While the synchronous nature of PPRC guarantees such ordering during periods of stability, certain types of rolling failure scenarios require additional consideration. By default, without the PPRC **consistency group** option enabled, if the primary ESS detects the loss of the physical PPRC connectivity or the failure of one of the secondary disks, it responds by moving the corresponding primary volumes to the suspended state but continues to process the subsequent I/O requests as normal. The subsystem does not report this condition to the driver and, as a result, the application continues normal processing without any knowledge of the lost updates. GPFS relies on log recovery techniques to provide for the atomicity of updates to the file system's metadata, which is why such behavior would expose GPFS to a serious data integrity risk. Therefore to provide for the proper ordering of updates to the recovery copy of the file system, it is suggested you always enable the **consistency group** option when configuring the PPRC paths between the peer logical subsystems.

Figure 8 on page 90 illustrates this problem. Consider a setup with two primary and two secondary subsystems, each providing access to two LUNs. The four primary LUNs make up the primary replica of the file system. At some point, GPFS attempts to issue a sequence of four write requests, one to each primary disk, with the expectation that the updates appear in the exact order they were issued. If PPRC

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 111

path 1 breaks before the start of the first write, the recovery site receives updates 3 and 4, but not necessarily 1 and 2 – a result that violates the write dependency rule and renders the target replica of the file system unusable.

The PPRC **consistency group** option determines the behavior of the primary subsystem in situations where a sudden failure of the secondary volume, or a failure of the inter-site interconnect, makes it impossible to sustain the normal synchronous mirroring process. If the PPRC path between the pair has been defined with the **consistency group** option, the primary volume enters a long busy state, and the subsystem reports this condition back to the host system with the QUEUE FULL (QF) SCSI status byte code. Typically, the driver makes several attempts to re-queue the request and ultimately reports the failure to the requestor. This allows GPFS to execute the appropriate action in response to the failure by either marking the disk down or panicking the file system. See the *GPFS: Problem Determination Guide*.

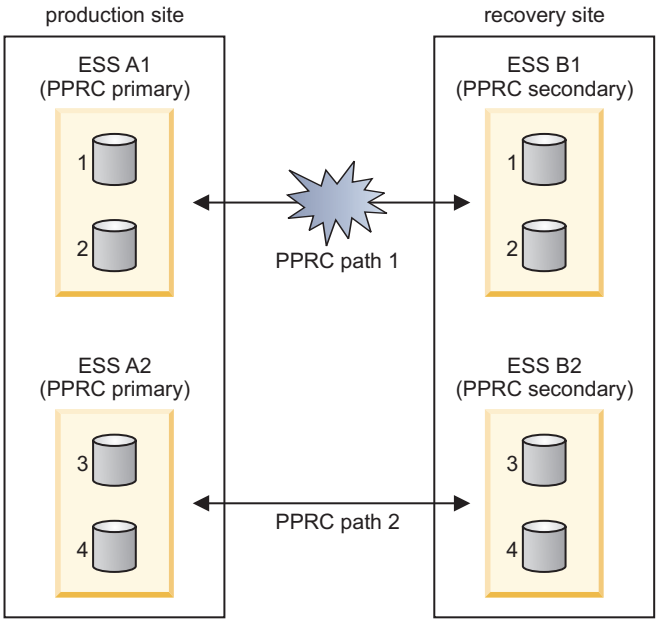


Figure 8. Violation of write ordering without the use of a PPRC consistency group

Asynchronous mirroring utilizing ESS FlashCopy

The FlashCopy feature of ESS provides an easy implementation to make a point-in-time copy of a GPFS file system as an online backup mechanism. This function provides an instantaneous copy of the original data on the target disk, while the actual transfer of data takes place asynchronously and is fully transparent to the user.

When a FlashCopy disk is first created, the subsystem establishes a control bitmap that is subsequently used to track the changes between the source and the target disks. When processing read I/O requests sent to the target disk, this bitmap is consulted to determine whether the request can be satisfied using the target's copy of the requested block. If the track containing the requested data has not yet been copied, the source disk is instead accessed and its copy of the data is used to satisfy the request. The FlashCopy feature is further described in the *IBM Enterprise Storage Server*. Similar to PPRC, FlashCopy operations can be invoked using the ESS Copy Services web interface or the command-line scripts.

To prevent the appearance of out-of-order updates, it is important to consider data consistency when using FlashCopy. Prior to taking the FlashCopy image all disk volumes that make up the file system must be brought to same logical point in time. Two methods may be used to provide for data consistency in the FlashCopy image of your GPFS file system. Both techniques guarantee the consistency of the

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 112

FlashCopy image by the means of temporary suspension of I/O, but either can be seen as the preferred method depending on your specific requirements and the nature of your GPFS client application:

- The use of FlashCopy consistency groups provides for the proper ordering of updates, but this method does not by itself suffice to guarantee the atomicity of updates as seen from the point of view of the user application. If the application process is actively writing data to GPFS, the on-disk content of the file system may, at any point in time, contain some number of incomplete data record updates and possibly some number of in-progress updates to the GPFS metadata. These appear as partial updates in the FlashCopy image of the file system, which must be dealt before enabling the image for normal file system use. The use of metadata logging techniques enables GPFS to detect and recover from these partial updates to the file system's metadata. However, ensuring the atomicity of updates to the actual data remains the responsibility of the user application. Consequently, the use of FlashCopy consistency groups is suitable only for applications that implement proper mechanisms for the recovery from incomplete updates to their data.

  The FlashCopy consistency group mechanism is used to *freeze* the source disk volume at the logical instant at which its image appears on the target disk:

  1. Issue the FlashCopy creation command, enabling the **freeze FlashCopy consistency group** option to suspend all I/O activity against each source volume in the file system. All I/O requests directed to the volumes now receive the SCSI status code **QUEUE FULL** (QF).
  2. Issue the **FlashCopy consistency created** task to release the consistency group and resume the normal processing of I/O. For details on the use of FlashCopy consistency groups, see the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.

  Assuming a configuration with:

  – Storage subsystems – ESS 1; logical subsystem LSS 1

  – LUN ids and disk volume names – **lunS1** (**hdisk11**), **lunS2** (**hdisk12**), **lunT1**, **lunT2**

    lunS1 and **lunS2** are the FlashCopy source volumes. These disks are SAN-connected and appear on the GPFS nodes as **hdisk11** and **hdisk12**, respectively. A single GPFS file system **fs0** has been defined over these two disks.

    lunT1 and **lunT2** are the FlashCopy target volumes. None of the GPFS nodes have direct connectivity to these disks.

  To generate a FlashCopy image using a consistency group:

  1. Run the **establish FlashCopy pair** task with the **freeze FlashCopy consistency group** option. Create the volume pairs:

     ```
     lunS1 – lunT1  (source-target)
     lunS2 – lunT2  (source-target)
     ```

     See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
  2. Run the **consistency created** task on all logical subsystems that hold any of the FlashCopy disks. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.

- The use of file-system-level suspension through the **mmfsctl** command prevents incomplete updates in the FlashCopy image and is the suggested method for protecting the integrity of your FlashCopy images. Issuing the **mmfsctl** command leaves the on-disk copy of the file system in a fully consistent state, ready to be flashed and copied onto a set of backup disks. The command instructs GPFS to flush the data buffers on all nodes, write the cached metadata structures to disk, and suspend the execution of all subsequent I/O requests.

  1. To initiate file-system-level suspension, issue the **mmfsctl suspend** command.
  2. To resume normal file system I/O, issue the **mmfsctl resume** command.

  Assuming a configuration with:

  – Storage subsystems – ESS 1; logical subsystem LSS 1

  – LUN ids and disk volume names – **lunS1** (**hdisk11**), **lunS2** (**hdisk12**), **lunT1**, **lunT2**

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 113

lunS1 and **lunS2** are the FlashCopy source volumes. These disks are SAN-connected and appear on the GPFS nodes as **hdisk11** and **hdisk12**, respectively. A single GPFS file system **fs0** has been defined over these two disks.

lunT1 and **lunT2** are the FlashCopy target volumes. None of the GPFS nodes have direct connectivity to these disks.

To generate a FlashCopy image using file-system-level suspension:

1. From any node in the GPFS cluster, suspend all file system activity and flush the GPFS buffers on all nodes:

   ```
   mmfsctl fs0 suspend
   ```

2. Run the **establish FlashCopy pair** task to create the following volume pairs:

   ```
   lunS1 – lunT1  (source-target)
   lunS2 – lunT2  (source-target)
   ```

   See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.

3. From any node in the GPFS cluster, resume the file system activity:

   ```
   mmfsctl fs0 resume
   ```

Several uses of the FlashCopy replica after its initial creation can be considered. For example, if your primary operating environment suffers a permanent loss or a corruption of data, you may choose to flash the target disks back onto the originals to quickly restore access to a copy of the file system as seen at the time of the previous snapshot. Before restoring the file system from a FlashCopy, please make sure to suspend the activity of the GPFS client processes and unmount the file system on all GPFS nodes.

To protect your data against site-wide disasters, you may chose to instead transfer the replica offsite to a remote location using PPRC or any other type of bulk data transfer technology. Alternatively, you may choose to mount the FlashCopy image as a separate file system on your backup processing server and transfer the data to tape storage. To enable regular file system access to your FlashCopy replica, create a single-node GPFS cluster on your backup server node and execute the **mmfsctl syncFSconfig** command to import the definition of the file system from your production GPFS cluster. Figure 9 on page 93 provides a schematic view of an asynchronous recovery GPFS environment using a combination of PPRC and FlashCopy. In such environments, two independent GPFS clusters are set up in distinct geographic locations (production and recovery sites). We refer to such clusters as *peer* GPFS clusters. Peer clusters must share the same UID/GID space, but otherwise need not belong to the same administrative domain. In particular, the administrator is free to identify nodes in both clusters with the same set of short hostnames if such a configuration is indeed desired.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 114

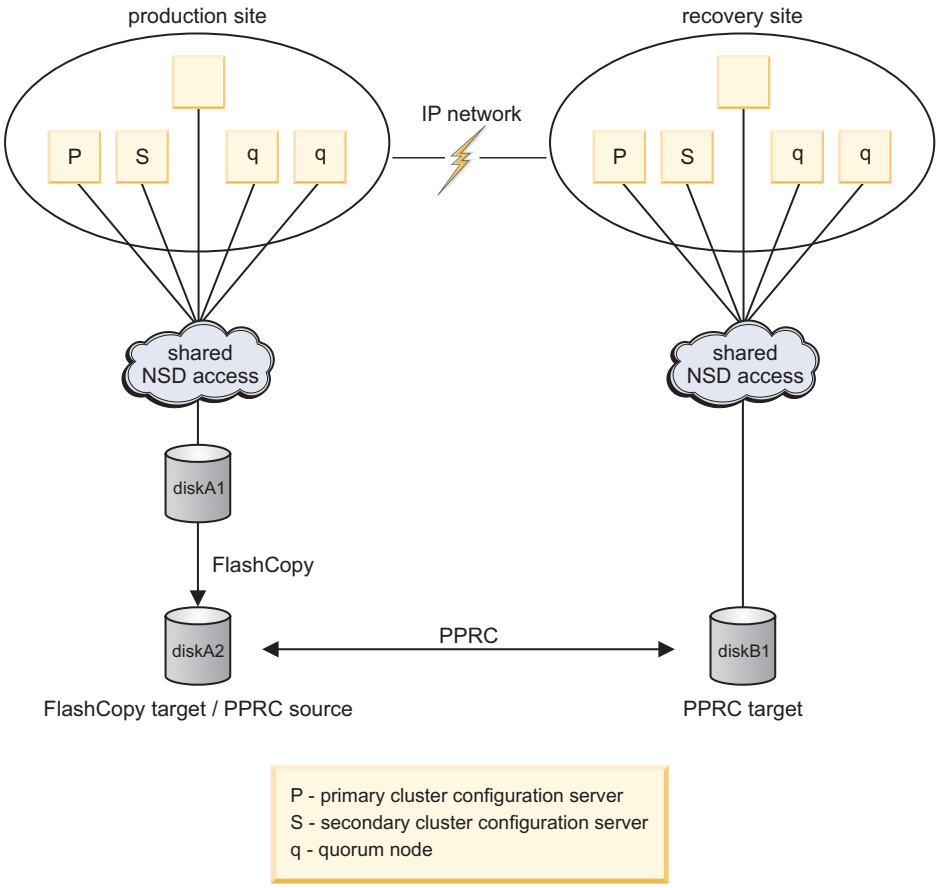


Figure 9. High-level organization of a FlashCopy/PPRC recovery environment

FlashCopy/PPRC provides for the availability of the file system's on-disk content in the recovery cluster. But in order to make the file system known and accessible, you must issue the **mmfsctl syncFSConfig** command to:

- Import the state of the file system's configuration from the primary location.
- Propagate all relevant changes to the configuration in the primary cluster to its peer to prevent the risks of discrepancy between the peer's **mmsdrfs** file and the content of the file system descriptor found in the snapshot.

  It is suggested you generate a new FlashCopy replica immediately after every administrative change to the state of the file system. This eliminates the risk of a discrepancy between the GPFS configuration data contained in the **mmsdrfs** file and the on-disk content of the replica.

Restriction: The primary copy of a GPFS file system and its FlashCopy image cannot coexist in the same GPFS cluster. A node can mount either the original copy of the file system or one of its FlashCopy replicas, but not both. This restriction has to do with the current implementation of the NSD-to-LUN mapping mechanism, which scans all locally-attached disks, searching for a specific value (the NSD id) at a particular location on disk. If both the original volume and its FlashCopy image are visible to a particular node, these disks would appear to GPFS as distinct devices with identical NSD ids

For this reason, we ask users to zone their SAN configurations such that at most one replica of any given GPFS disk is visible from any node. That is, the nodes in your production cluster should have access to the disks that make up the actual file system but should not see the disks holding the FlashCopy images, whereas the backup server should see the FlashCopy targets but not the originals.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 115

Alternatively, you can use the **nsddevices** user exit located in **/var/mmfs/etc/** to explicitly define the subset of the locally visible disks to be accessed during the NSD device scan on the local node. See "Setting up FlashCopy using file-system-level suspension."

In the production GPFS cluster, FlashCopy is used to take periodic volume-level snapshots of the GPFS file system onto a set of idle local disks and the snapshots are then propagated to the peer recovery cluster using PPRC. The goal of this technique is to provide a consistent (but not necessarily up-to-date) image of the file system at the recovery site that can be used to restore operation in the event of a disaster in the primary cluster. Note that since from the GPFS perspective, the replicas are two entirely distinct file systems, nothing prevents the administrator from mounting and operating on both replicas concurrently if deemed necessary.

Setting up FlashCopy using file-system-level suspension

To prepare a file system as depicted in Figure 9 on page 93, using file-system-level suspension to provide for data consistency, consider the configuration:

Site A - primary cluster
Consisting of:

- Nodes – **nodeA001**, **nodeA002**, **nodeA003**, **nodeA004**, **nodeA005**
- Disk device names – **diskA1**, **diskA2**

Site B - recovery site
Consisting of:

- Nodes – **nodeB001**, **nodeB002**, **nodeB003**, **nodeB004**, **nodeB005**
- Disks – **diskB1**

There is a single file system, **fs0**, defined on **diskA1**. To create a volume-level snapshot and propagate the snapshot to the recovery cluster:

1. Define an **nsddevices** user exit file to prevent the production site from using the FlashCopy target disk **diskA2**:

   ```
   echo "echo diskA1 hdisk" > /var/mmfs/etc/nsddevices
   chmod 744 /var/mmfs/etc/nsddevices
   ```

   Refer to the prolog of **/usr/lpp/mmfs/samples/nsddevices.samples** for detailed instructions on the usage of **nsddevices**.
2. In the primary cluster, suspend all file system I/O activity and flush the GPFS buffers:

   ```
   mmfsctl fs0 suspend
   ```
3. Establish a FlashCopy pair using **diskA1** as the source and **diskA2** as the target. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
4. Resume the file system I/O activity:

   ```
   mmfsctl fs0 resume
   ```
5. Establish a PPRC path and a synchronous PPRC volume pair **diskA2-diskB** (primary-secondary). Use the **copy entire volume** option and leave the **permit read from secondary** option disabled. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
6. Wait for the completion of the FlashCopy background task. Wait for the PPRC pair to reach the *duplex* (fully synchronized) state.
7. Terminate the PPRC volume pair **diskA2-diskB**. See the *IBM Enterprise Storage Server Implementing ESS Copy Services in Open Environments*.
8. If this is the first time the snapshot is taken, or if the configuration state of **fs0** changed since the previous snapshot, propagate the most recent configuration to site **B**:

   ```
   mmfsctl all syncFSconfig -n nodes.siteB
   ```

   The file **nodes.siteB** lists all of the nodes, one per line, at the recovery site.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 116

Chapter 7. Implementing a clustered NFS using GPFS on Linux

In addition to the traditional exporting of GPFS file systems using the Network File System (NFS) protocol, GPFS allows you to configure a subset of the nodes in the cluster to provide a highly-available solution for exporting GPFS file systems using NFS.

Note: This feature is available with GPFS Standard Edition or higher.

The participating nodes are designated as Cluster NFS (CNFS) member nodes and the entire setup is frequently referred to as CNFS or a CNFS cluster.

In this solution, all CNFS nodes export the same file systems to the NFS clients. When one of the CNFS nodes fails, the NFS serving load moves from the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover. For the NFS client node to experience a seamless failover, hard mounts must be used. The use of soft mounts will likely result in stale NFS file handle conditions when a server experiences a problem, even though CNFS failover will still be done.

Currently, CNFS is supported only in the Linux environment. For an up-to-date list of supported operating systems, specific distributions, and other dependencies, refer to the GPFS FAQ in the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) or GPFS FAQ in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfsclustersfaq.html).

NFS monitoring

Every node in the CNFS cluster runs a separate GPFS utility that monitors GPFS, NFS, and networking components on the node. Upon failure detection and based on your configuration, the monitoring utility might invoke a failover.

NFS failover

As part of GPFS recovery, the CNFS cluster failover mechanism is invoked. It transfers the NFS serving load that was served by the failing node to another node in the CNFS cluster. Failover is done using recovery groups to help choose the preferred node for takeover.

The failover mechanism is based on IP address failover. The CNFS IP address is moved from the failing node to a healthy node in the CNFS cluster. In addition, it guarantees NFS lock (NLM) recovery.

Failover processing may involve rebooting of the problem node. To minimize the effects of the reboot, it is recommended that the CNFS nodes are dedicated to that purpose and are not used to run other critical processes. CNFS node rebooting should not be disabled or the failover reliability will be severely impacted.

NFS locking and load balancing

CNFS supports a failover of all of the node's load together (all of its NFS IP addresses) as one unit to another node. However, if no locks are outstanding, individual IP addresses can be moved to other nodes for load balancing purposes.

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 117

CNFS is dependent on DNS for any automated load balancing of NFS clients among the NFS cluster nodes. Using the round-robin algorithm is highly recommended.

CNFS network setup

In addition to one set of IP addresses for the GPFS cluster, a separate set of one or more IP addresses is required for CNFS serving.

While the GPFS cluster may be defined over an IPv6 network, the IP addresses specified for CNFS must be IPv4.

CNFS setup

You can set up a clustered NFS environment within a GPFS cluster.

To do this, follow these steps:

1. Designate a separate directory for the CNFS shared files:

   mmchconfig cnfsSharedRoot=*directory*

   where:

   cnfsSharedRoot=*directory*
   Is the path name to a GPFS directory, preferably on a small separate file system that is not exported by NFS. The GPFS file system that contains the directory must be configured to be mounted automatically upon GPFS start on all of the CNFS nodes (**-A yes** option on the **mmchfs** command). **cnfsSharedRoot** is a mandatory parameter and must be defined first.
2. Add all GPFS file systems that need to be exported to **/etc/exports**. See *Exporting a GPFS file system using NFS* in the *GPFS: Administration and Programming Reference* for NFS export considerations. If the shared directory from step 1 is in an exported file system, restrict access to that directory.
3. Use the **mmchnode** command to add nodes to the CNFS cluster:

   mmchnode --cnfs-interface=*nfs_ip_address_list* **-N** *node*

   where:

   nfs_ip_address_list
   Is a comma-separated list of host names or IP addresses to be used for GPFS cluster NFS serving.

   node Identifies a GPFS node to be added to the CNFS cluster.

   See the description of the **mmchnode** command in the *GPFS: Administration and Programming Reference* for information about how to use the command with a list of nodes.
4. Use the **mmchconfig** command to configure the optional CNFS parameters.

   cnfsMountdPort=*mountd_port*
   Specifies the port number to be used for the **rpc.mountd** daemon.

   For CNFS to work correctly with the automounter (AMD), the **rpc.mountd** daemon on the different nodes must be bound to the same port.

   cnfsNFSDprocs=*nfsd_procs*
   Specifies the number of **nfsd** kernel threads. The default is 32.
5. If multiple failover groups are desired, assign a group ID to each NFS node:

   mmchnode --cnfs-groupid=*nn* **-N** *node*

   To assign NFS nodes to different groups, use a group ID that is in a different range of ten. For example, a node with group ID 2*n* will fail over only to nodes in the same range of ten (which means any node with group ID 20 to 29). Failover in the same group will first look for one of the nodes with the same group ID. If none are found, any node in the group range starting at *n*0 to *n*9 is selected.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 118

CNFS administration

There are some common CNFS administration tasks in this topic along with a sample configuration.

To query the current CNFS configuration, enter:

```
mmlscluster --cnfs
```

To temporarily disable CNFS on one or more nodes, enter:

```
mmchnode --cnfs-disable -N NodeList
```

Note: This operation affects only the high-availability aspects of the CNFS functionality. Normal NFS exporting of the data from the node is not affected. All currently defined CNFS IP addresses remain unchanged. There will be no automatic failover from or to this node in case of a failure. If failover is desired, GPFS should be shut down on the affected node prior to issuing the **mmchnode** command.

To re-enable previously-disabled CNFS member nodes, enter:

```
mmchnode --cnfs-enable -N NodeList
```

Note: If the GPFS daemon is running on a node on which CNFS is being re-enabled, the node will try to activate its CNFS IP address. That activation would include a takeover, if the IP address was currently on some other CNFS-enabled node.

To permanently remove nodes from the CNFS cluster, enter:

```
mmchnode --cnfs-interface=DELETE -N NodeList
```

Note: This operation affects only the high-availability aspects of the CNFS functionality. Normal NFS exporting of the data from the node is not affected. All currently defined CNFS IP addresses remain unchanged. There will be no automatic failover from or to this node in case of a failure. If failover is desired, GPFS should be shut down on the affected node prior to issuing the **mmchnode** command.

A sample CNFS configuration

Here is a CNFS configuration example, which assumes the following:

- Your GPFS cluster contains 3 nodes: fin18, fin19, and fin20
- The IP addresses for NFS serving are: fin18nfs, fin19nfs, and fin20nfs

To define a CNFS cluster made up of these nodes, follow these steps:

1. Add the desired GPFS file systems to **/etc/exports** on each of the nodes.
2. Create a directory called **ha** in one of the GPFS file systems by entering:
   ```
   mkdir /gpfs/fs1/ha
   ```
3. Create a temporary file called **/tmp/hanfs-list**, which contains the following lines:
   ```
   fin18 --cnfs-interface=fin18nfs
   fin19 --cnfs-interface=fin19nfs
   fin20 --cnfs-interface=fin20nfs
   ```
4. Set the CNFS shared directory by entering:
   ```
   mmchconfig cnfsSharedRoot=/gpfs/fs1/ha
   ```
5. Create the CNFS cluster with the **mmchnode** command, by entering:
   ```
   mmchnode -S /tmp/hanfs-list
   ```
6. Access the exported GPFS file systems over NFS. If one or more GPFS nodes fail, the NFS clients should continue uninterrupted.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 119

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 120

Chapter 8. Performance and health monitoring

GPFS provides system administrators with the ability to monitor GPFS performance and health.

Performance and health topics include:

- "Network performance monitoring"

Network performance monitoring

Network performance can be monitored with Remote Procedure Call (RPC) statistics.

The GPFS daemon caches statistics relating to RPCs. Most statistics are related to RPCs sent to other nodes. This includes a set of up to seven statistics cached per node and one statistic that is cached per size of the RPC message. For RPCs received from other nodes, one statistic is cached for each type of RPC message.

The statistics cached per node are the following:

Channel wait time
: The amount of time the RPC must wait for access to a communication channel to the target node.

Send time TCP
: The amount of time to transfer an RPC message to an Ethernet interface.

Send time verbs
: The amount of time to transfer an RPC message to an InfiniBand interface.

Receive time TCP
: The amount of time to transfer an RPC message from an Ethernet interface into the daemon.

Latency TCP
: The latency of the RPC when sent and received over an Ethernet interface.

Latency verbs
: The latency of the RPC when sent and received over an InfiniBand interface.

Latency mixed
: The latency of the RPC when sent over one type of interface (Ethernet or InfiniBand) and received over the other (InfiniBand or Ethernet).

If an InfiniBand network is not configured, no statistics are cached for send time verbs, latency verbs, and latency mixed.

The latency of an RPC is defined as the round-trip time minus the execution time on the target node. The round-trip time is measured from the start of writing the RPC message to the interface until the RPC reply is completely received. The execution time is measured on the target node from the time the message is completely received until the time the reply is sent. The latency, therefore, is the amount of time the RPC is being transmitted and received over the network and is a relative measure of the network performance as seen by the GPFS daemon.

There is a statistic associated with each of a set of size ranges, each with an upper bound that is a power of 2. The first range is 0 through 64, then 65 through 128, then 129 through 256, and then continuing until the last range has an upper bound of twice the **maxBlockSize**. For example, if the **maxBlockSize** is 1 MB, the upper bound of the last range is 2,097,152 (2 MB). For each of these ranges, the associated statistic is the latency of the RPC whose size falls within that range. The size of an RPC is the amount of data sent

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 121

plus the amount of data received. However, if one amount is more than 16 times greater than the other, only the larger amount is used as the size of the RPC.

The final statistic associated with each type of RPC message, on the node where the RPC is received, is the execution time of the RPC.

Each of the statistics described so far is actually an aggregation of values. By default, an aggregation consists of 60 one-second intervals, 60 one-minute intervals, 24 one-hour intervals, and 30 one-day intervals. Each interval consists of a sum of values accumulated during the interval, a count of values added into the sum, the minimum value added into the sum, and the maximum value added into the sum. Sixty seconds after the daemon starts, each of the one-second intervals contains data and every second thereafter the oldest interval is discarded and a new one entered. An analogous pattern holds for the minute, hour, and day periods.

As each RPC reply is received, the following information is saved in a *raw statistics buffer*:

- channel wait time
- send time
- receive time
- latency
- length of data sent
- length of data received
- flags indicating if the RPC was sent or received over InfiniBand
- target node identifier

As each RPC completes execution, the execution time for the RPC and the message type of the RPC is saved in a *raw execution buffer*. Once per second these raw buffers are processed and the values are added to the appropriate aggregated statistic. For each value, the value is added to the statistic's sum, the count is incremented, and the value is compared to the minimum and maximum, which are adjusted as appropriate. Upon completion of this processing, for each statistic the sum, count, minimum, and maximum values are entered into the next one-second interval.

Every 60 seconds, the sums and counts in the 60 one-second intervals are added into a one-minute sum and count. The smallest of the 60 minimum values is determined, and the largest of the 60 maximum values is determined. This one-minute sum, count, minimum, and maximum are then entered into the next one-minute interval.

An analogous pattern holds for the minute, hour, and day periods. For any one particular interval, the sum is the sum of all raw values processed during that interval, the count is the count of all values during that interval, the minimum is the minimum of all values during that interval, and the maximum is the maximum of all values during that interval.

When statistics are displayed for any particular interval, an average is calculated from the sum and count, then the average, minimum, maximum, and count are displayed. The average, minimum and maximum are displayed in units of milliseconds, to three decimal places (one microsecond granularity).

The following **mmchconfig** attributes are available to control the RPC buffers and intervals:

- **rpcPerfRawStatBufferSize**
- **rpcPerfRawExecBufferSize**
- **rpcPerfNumberSecondIntervals**
- **rpcPerfNumberMinuteIntervals**
- **rpcPerfNumberHourIntervals**
- **rpcPerfNumberDayIntervals**

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 122

The **mmdiag** command with the **--rpc** parameter can be used to query RPC statistics.

The **mmchconfig** and **mmdiag** commands are fully described in the *Commands* topic in the *GPFS: Administration and Programming Reference*.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 123

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 124

Chapter 9. Monitoring GPFS I/O performance with the mmpmon command

Use the **mmpmon** command to monitor GPFS performance on the node in which it is run, and other specified nodes.

Before attempting to use the **mmpmon** command, review the command documentation in the *GPFS: Administration and Programming Reference*, and then read all the following relevant **mmpmon** topics.

- "Overview of mmpmon"
- "Specifying input to the mmpmon command"
- "Example mmpmon scenarios and how to analyze and interpret their results" on page 126
- "Other information about mmpmon output" on page 134

Overview of mmpmon

The **mmpmon** facility allows the system administrator to collect I/O statistics from the point of view of GPFS servicing application I/O requests.

The collected data can be used for many purposes, including:

- Tracking IO demand over longer periods of time - weeks or months.
- Recording I/O patterns over time (when peak usage occurs, and so forth).
- Determining if some nodes service more application demand than others.
- Monitoring the I/O patterns of a single application which is spread across multiple nodes.
- Recording application I/O request service times.

Figure 10 shows the software layers in a typical system with GPFS. **mmpmon** is built into GPFS.

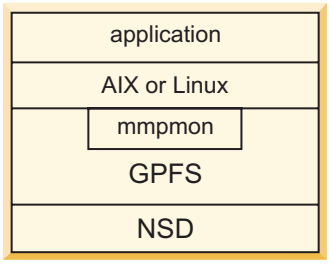


Figure 10. Node running mmpmon

Specifying input to the mmpmon command

The **mmpmon** command must be run using root authority. For command syntax, see **mmpmon** in the *GPFS: Administration and Programming Reference*.

The **mmpmon** command is controlled by an input file that contains a series of requests, one per line. This input can be specified with the **-i** flag, or read from standard input (stdin). Providing input using stdin allows **mmpmon** to take keyboard input or output piped from a user script or application.

Leading blanks in the input file are ignored. A line beginning with a pound sign (#) is treated as a comment. Leading blanks in a line whose first non-blank character is a pound sign (#) are ignored.

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032

Table 5 describes the **mmpmon** input requests.

Table 5. Input requests to the **mmpmon** *command*

Request	Description
fs_io_s	"Display I/O statistics per mounted file system"
io_s	"Display I/O statistics for the entire node" on page 106
nlist add *name*[*name*...]	"Add node names to a list of nodes for mmpmon processing" on page 108
nlist del	"Delete a node list" on page 109
nlist new *name*[*name*...]	"Create a new node list" on page 110
nlist s	"Show the contents of the current node list" on page 110
nlist sub *name*[*name*...]	"Delete node names from a list of nodes for mmpmon processing" on page 111
once *request*	Indicates that the request is to be performed only once.
reset	"Reset statistics to zero" on page 107
rhist nr	"Changing the request histogram facility request size and latency ranges" on page 117
rhist off	"Disabling the request histogram facility" on page 118. This is the default.
rhist on	"Enabling the request histogram facility" on page 119
rhist p	"Displaying the request histogram facility pattern" on page 120
rhist reset	"Resetting the request histogram facility data to zero" on page 122
rhist s	"Displaying the request histogram facility statistics values" on page 123
source *filename*	"Using request *source* and prefix directive *once*" on page 129
ver	"Displaying mmpmon version" on page 125

Running mmpmon on multiple nodes

The **mmpmon** command may be invoked on one node to submit requests to multiple nodes in a local GPFS cluster by using the **nlist** requests. See "Understanding the node list facility" on page 107.

Running mmpmon concurrently from multiple users on the same node

Five instances of **mmpmon** may be run on a given node concurrently. This is intended primarily to allow different user-written performance analysis applications or scripts to work with the performance data. For example, one analysis application might deal with **fs_io_s** and **io_s** data, while another one deals with **rhist** data, and another gathers data from other nodes in the cluster. The applications might be separately written or separately maintained, or have different sleep and wake-up schedules.

Be aware that there is only one set of counters for **fs_io_s** and **io_s** data, and another, separate set for **rhist** data. Multiple analysis applications dealing with the same set of data must coordinate any activities that could reset the counters, or in the case of **rhist** requests, disable the feature or modify the ranges.

Display I/O statistics per mounted file system

The **fs_io_s** (file system I/O statistics) request returns strings containing I/O statistics taken over all mounted file systems as seen by that node, and are presented as total values for each file system. The values are cumulative since the file systems were mounted or since the last **reset** request, whichever is most recent. When a file system is unmounted, its statistics are lost.

Read and write statistics are recorded separately. The statistics for a given file system are for the file system activity on the node running **mmpmon**, not the file system in total (across the cluster).

Table 6 on page 105 describes the keywords for the **fs_io_s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 126

Table 6. Keywords and values for the **mmpmon fs_io_s** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_cl_	Name of the cluster that owns the file system.
_fs_	The name of the file system for which data are being presented.
_d_	The number of disks in the file system.
_br_	Total number of bytes read, from both disk and cache.
_bw_	Total number of bytes written, to both disk and cache.
_oc_	Count of **open()** call requests serviced by GPFS. This also includes **creat()** call counts.
_cc_	Number of **close()** call requests serviced by GPFS.
rdc	Number of application read requests serviced by GPFS.
_wc_	Number of application write requests serviced by GPFS.
dir	Number of **readdir()** call requests serviced by GPFS.
_iu_	Number of inode updates to disk.

Example of mmpmon fs_io_s request

Assume that **commandFile** contains this line:

```
fs_io_s
```

and this command is issued:

```
mmpmon -p  -i commandFile
```

The output is two lines in total, and similar to this:

```
_fs_io_s_ _n_ 1992.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407431 _cl_ myCluster.xxx.com
_fs_ gpfs2 _d_ 2 _br_ 6291456 _bw_ 314572800 _oc_ 10 _cc_ 16 _rdc_ 101 _wc_ 300 _dir_ 7 _iu_ 2
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407455 _cl_ myCluster.xxx.com
_fs_ gpfs1 _d_ 3 _br_ 5431636 _bw_ 173342800 _oc_ 6 _cc_ 8 _rdc_ 54 _wc_ 156 _dir_ 3 _iu_ 6
```

The output consists of one string per mounted file system. In this example, there are two mounted file systems, **gpfs1** and **gpfs2**.

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:    myCluster.xxx.com
filesystem: gpfs2
disks: 2
timestamp: 1066660148/407431
bytes read: 6291456
bytes written: 314572800
opens: 10
closes: 16
reads: 101
writes: 300
readdir: 7
inode updates: 2

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:    myCluster.xxx.com
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 127

```
filesystem: gpfs1
disks: 3
timestamp: 1066660148/407455
bytes read: 5431636
bytes written: 173342800
opens: 6
closes: 8
reads: 54
writes: 156
readdir: 3
inode updates: 6
```

When no file systems are mounted, the responses are similar to:

```
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 1 _t_ 1066660148 _tu_ 407431 _cl_ - _fs_ -
```

The **_rc_** field is nonzero and the both the **_fs_** and **_cl_** fields contains a minus sign. If the **-p** flag is not specified, the results are similar to:

```
mmpmon node 199.18.1.8 name node1 fs_io_s status 1
no file systems mounted
```

Display I/O statistics for the entire node

The **io_s** (I/O statistics) request returns strings containing I/O statistics taken over all mounted file systems as seen by that node, and are presented as total values for the entire node. The values are cumulative since the file systems were mounted or since the last **reset**, whichever is most recent. When a file system is unmounted, its statistics are lost and its contribution to the total node statistics vanishes. Read and write statistics are recorded separately.

Table 7 describes the keywords for the **io_s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 7. Keywords and values for the **mmpmon io_s** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_br_	Total number of bytes read, from both disk and cache.
_bw_	Total number of bytes written, to both disk and cache.
_oc_	Count of **open()** call requests serviced by GPFS. The open count also includes **creat()** call counts.
_cc_	Number of **close()** call requests serviced by GPFS.
rdc	Number of application read requests serviced by GPFS.
_wc_	Number of application write requests serviced by GPFS.
dir	Number of **readdir()** call requests serviced by GPFS.
_iu_	Number of inode updates to disk. This includes inodes flushed to disk because of access time updates.

Example of mmpmon io_s request

Assume that **commandFile** contains this line:

```
io_s
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 128

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is one line in total, and similar to this:

```
_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407431 _br_ 6291456
_bw_ 314572800 _oc_ 10 _cc_ 16 _rdc_ 101 _wc_ 300 _dir_ 7 _iu_ 2
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 io_s OK
timestamp: 1066660148/407431
bytes read: 6291456
bytes written: 314572800
opens: 10
closes: 16
reads: 101
writes: 300
readdir: 7
inode updates: 2
```

Reset statistics to zero

The **reset** request resets the statistics that are displayed with **fs_io_s** and **io_s** requests. The **reset** request *does not* reset the histogram data, which is controlled and displayed with **rhist** requests. Table 8 describes the keywords for the **reset** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag. The response is a single string.

Table 8. Keywords and values for the **mmpmon reset** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Example of mmpmon reset request

Assume that **commandFile** contains this line:

```
reset
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_reset_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1066660148 _tu_ 407431
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 reset OK
```

Understanding the node list facility

The node list facility can be used to invoke **mmpmon** on multiple nodes and gather data from other nodes in the cluster. Table 9 on page 108 describes the **nlist** requests:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 129

Table 9. nlist requests for the **mmpmon** *command*

Request	Description
nlist add *name*[*name*...]	"Add node names to a list of nodes for mmpmon processing"
nlist del	"Delete a node list" on page 109
nlist new *name*[*name*...]	"Create a new node list" on page 110
nlist s	"Show the contents of the current node list" on page 110
nlist sub *name*[*name*...]	"Delete node names from a list of nodes for mmpmon processing" on page 111

When specifying node names, keep these points in mind:

1. A node name of '.' (dot) indicates the current node.
2. A node name of '*' (asterisk) indicates all currently connected local cluster nodes.
3. The nodes named in the node list must belong to the local cluster. Nodes in remote clusters are not supported.
4. A node list can contain nodes that are currently down. When an inactive node comes up, **mmpmon** will attempt to gather data from it.
5. If a node list contains an incorrect or unrecognized node name, all other entries in the list are processed. Suitable messages are issued for an incorrect node name.
6. When **mmpmon** gathers responses from the nodes in a node list, the full response from one node is presented before the next node. Data is not interleaved. There is no guarantee of the order of node responses.
7. The node that issues the **mmpmon** command need not appear in the node list. The case of this node serving only as a collection point for data from other nodes is a valid configuration.

Add node names to a list of nodes for mmpmon processing

The **nlist add** (node list add) request is used to add node names to a list of nodes for **mmpmon** to collect their data. The node names are separated by blanks.

Table 10 describes the keywords for the **nlist add** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 10. Keywords and values for the **mmpmon nlist add** *response*

Keyword	Description
_n_	IP address of the node processing the node list. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `add`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_c_	The number of nodes in the user-supplied list.
_ni_	Node name input. A user-supplied node name from the offered list of names.
_nx_	Node name translation. The preferred GPFS name for the node.
nxip	Node name translated IP address. The preferred GPFS IP address for the node.
did	The number of nodes names considered valid and processed by the requests.
nlc	The number of nodes in the node list now (after all processing).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 130

If the **nlist add** request is issued when no node list exists, it is handled as if it were an **nlist new** request.

Example of mmpmon nlist add request

A two- node cluster has nodes **node1** (199.18.1.2), a non-quorum node, and **node2** (199.18.1.5), a quorum node. A remote cluster has node **node3** (199.18.1.8). The **mmpmon** command is run on **node1**.

Assume that **commandFile** contains this line:

```
nlist add n2 199.18.1.2
```

and this command is issued:

```
mmpmon -p -i commandFile
```

Note in this example that an alias name **n2** was used for **node2**, and an IP address was used for **node1**. Notice how the values for **_ni_** and **_nx_** differ in these cases.

The output is similar to this:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _c_ 2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _ni_ n2 _nx_
node2 _nxip_ 199.18.1.5
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _ni_
199.18.1.2 _nx_ node1 _nxip_ 199.18.1.2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ add _rc_ 0 _t_ 1121955894 _tu_ 261881 _did_ 2 _nlc_
2
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist add
initial status 0
name count 2
timestamp 1121955879/468858
node name n2, OK (name used: node2, IP address 199.18.1.5)
node name 199.18.1.2, OK (name used: node1, IP address 199.18.1.2)
final status 0
node names processed 2
current node list count 2
```

The requests **nlist add** and **nlist sub** behave in a similar way and use the same keyword and response format.

These requests are rejected if issued while quorum has been lost.

Delete a node list

The **nlist del** (node list delete) request deletes a node list, if one exists. If no node list exists, the request succeeds and no error code is produced.

Table 11 describes the keywords for the **nlist del** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 11. Keywords and values for the **mmpmon nlist del** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `del`.
_rc_	Indicates the status of the operation.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 131

Table 11. Keywords and values for the **mmpmon nlist del** *response (continued)*

Keyword	Description
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Example of mmpmon nlist del request

Assume that **commandFile** contains this line:

```
nlist del
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ del _rc_ 0 _t_ 1121956817 _tu_ 46050
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist del status OK timestamp 1121956908/396381
```

Create a new node list

The **nlist new** (node list new) request deletes the current node list if one exists, creates a new, empty node list, and then attempts to add the specified node names to the node list. The node names are separated by blanks.

Table 12 describes the keywords for the **nlist new** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 12. Keywords and values for the **mmpmon nlist new** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `new`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Show the contents of the current node list

The **nlist s** (node list show) request displays the current contents of the node list. If no node list exists, a count of zero is returned and no error is produced.

Table 13 describes the keywords for the **nlist s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 13. Keywords and values for the **mmpmon nlist s** *response*

Keyword	Description
_n_	IP address of the node processing the request. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `s`.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 132

Table 13. Keywords and values for the **mmpmon nlist s** *response (continued)*

Keyword	Description
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_c_	Number of nodes in the node list.
mbr	GPFS preferred node name for the list member.
_ip_	GPFS preferred IP address for the list member.

Example of mmpmon nlist s request

Assume that **commandFile** contains this line:

```
nlist s
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121956950 _tu_ 863292 _c_ 2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121956950 _tu_ 863292 _mbr_ node1
_ip_ 199.18.1.2
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121956950 _tu_ 863292 _mbr_
node2 _ip_ 199.18.1.5
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist s
status 0
name count 2
timestamp 1121957505/165931
node name node1, IP address 199.18.1.2
node name node2, IP address 199.18.1.5
```

If there is no node list, the response looks like:

```
_nlist_ _n_ 199.18.1.2 _nn_ node1 _req_ s _rc_ 0 _t_ 1121957395 _tu_ 910440 _c_ 0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 nlist s
status 0
name count 0
timestamp 1121957436/353352
the node list is empty
```

The **nlist s** request is rejected if issued while quorum has been lost. Only one response line is presented.

```
_failed_ _n_ 199.18.1.8 _nn_ node2 _rc_ 668 _t_ 1121957395 _tu_ 910440
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node2: failure status 668 timestamp 1121957395/910440
lost quorum
```

Delete node names from a list of nodes for mmpmon processing

The **nlist sub** (subtract a node from the node list) request removes a node from a list of node names. This keywords and responses are similar to the **nlist add** request. The **_req_** keyword (action requested) for **nlist sub** is sub.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 133

Node list examples and error handling

The **nlist** facility can be used to obtain GPFS performance data from nodes other than the one on which the **mmpmon** command is invoked. This information is useful to see the flow of GPFS I/O from one node to another, and spot potential problems.

A successful fs_io_s request propagated to two nodes

In this example, an **fs_io_s** request is successfully propagated to two nodes. This command is issued:

```
mmpmon -p -i command_file
```

where **command_file** has this:

```
nlist new node1 node2
fs_io_s
```

The output is similar to this:

```
_fs_io_s_ _n_ 199.18.1.2 _nn_ node1 _rc_ 0 _t_ 1121974197 _tu_ 278619 _cl_
xxx.localdomain _fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
_dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.2 _nn_ node1 _rc_ 0 _t_ 1121974197 _tu_ 278619 _cl_
xxx.localdomain _fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
_dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974167 _tu_ 116443 _cl_
cl1.xxx.com _fs_ fs3 _d_ 3 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 3
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974167 _tu_ 116443 _cl_
cl1.xxx.comm _fs_ fs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974167 _tu_ 116443 _cl_
xxx.localdomain _fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
_dir_ 0 _iu_ 0
```

The responses from a propagated request are the same as they would have been if issued on each node separately.

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1121974088/463102
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.2 name node1 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs1
disks: 1
timestamp: 1121974088/463102
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 134

```
mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs3
disks: 3
timestamp: 1121974058/321741
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 2

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs2
disks: 2
timestamp: 1121974058/321741
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1121974058/321741
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0
```

Failure on a node accessed by mmpmon

In this example, the same scenario is run on **node2**, but with a failure on **node1** (a non-quorum node) because **node1** was shutdown:

```
_failed_ _n_ 199.18.1.5 _nn_ node2 _fn_ 199.18.1.2 _fnn_ node1 _rc_ 233
_t_ 1121974459 _tu_ 602231
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974459 _tu_ 616867 _cl_
cl1.xxx.com _fs_ fs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974459 _tu_ 616867 _cl_
cl1.xxx.com _fs_ fs3 _d_ 3 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0
_iu_ 0
_fs_io_s_ _n_ 199.18.1.5 _nn_ node2 _rc_ 0 _t_ 1121974459 _tu_ 616867 _cl_
node1.localdomain _fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.5 name node2:
from node 199.18.1.2 from name node1: failure status 233 timestamp 1121974459/602231
node failed (or never started)
mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs2
disks: 2
timestamp: 1121974544/222514
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 135

```
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: cl1.xxx.com
filesystem: fs3
disks: 3
timestamp: 1121974544/222514
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0

mmpmon node 199.18.1.5 name node2 fs_io_s OK
cluster: xxx.localdomain
filesystem: gpfs2
disks: 2
timestamp: 1121974544/222514
bytes read: 0
bytes written: 0
opens: 0
closes: 0
reads: 0
writes: 0
readdir: 0
inode updates: 0
```

Node shutdown and quorum loss

In this example, the quorum node (**node2**) is shutdown, causing quorum loss on **node1**. Running the same example on **node2**, the output is similar to:

```
_failed_ _n_ 199.18.1.2 _nn_ node1 _rc_ 668 _t_ 1121974459 _tu_ 616867
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.2 name node1: failure status 668 timestamp 1121974459/616867
lost quorum
```

In this scenario there can be a window where **node2** is down and **node1** has not yet lost quorum. When quorum loss occurs, the **mmpmon** command does not attempt to communicate with any nodes in the node list. The goal with failure handling is to accurately maintain the node list across node failures, so that when nodes come back up they again contribute to the aggregated responses.

Node list failure values

Table 14 describes the keywords and values produced by the **mmpmon** command on a node list failure:

Table 14. Keywords and values for the **mmpmon nlist** *failures*

Keyword	Description
_n_	IP address of the node processing the node list. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_fn_	IP address of the node that is no longer responding to **mmpmon** requests.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 136

Table 14. Keywords and values for the **mmpmon nlist** *failures (continued)*

Keyword	Description
fnn	The name by which GPFS knows the node that is no longer responding to **mmpmon** requests
_rc_	Indicates the status of the operation. See "Return codes from mmpmon" on page 135.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Understanding the request histogram facility

The **mmpmon** requests whose name starts with **rhist** control the request histogram facility. This facility tallies I/O operations using a set of counters. Counters for reads and writes are kept separately. They are categorized according to a pattern that may be customized by the user. A default pattern is also provided. The **size range** and **latency range** input parameters to the **rhist nr** request are used to define the pattern.

The first time that you run the **rhist** requests, assess if there is a noticeable performance degradation. Collecting histogram data may cause performance degradation. This is possible once the histogram facility is enabled, but will probably not be noticed while the commands themselves are running. It is more of a long term issue as the GPFS daemon runs with histograms enabled.

The histogram lock is used to prevent two **rhist** requests from being processed simultaneously. If an **rhist** request fails with an **_rc_** of 16, the lock is in use. Reissue the request.

The histogram data survives file system mounts and unmounts. In order to reset this data, use the **rhist reset** request.

Specifying the size ranges for I/O histograms

The size ranges are used to categorize the I/O according to the size, in bytes, of the I/O operation. The size ranges are specified using a string of positive integers separated by semicolons (;). No white space is allowed within the size range operand. Each number represents the upper bound, in bytes, of the I/O request size for that range. The numbers must be monotonically increasing. Each number may be optionally followed by the letters K or k to denote multiplication by 1024, or by the letters M or m to denote multiplication by 1048576 (1024*1024).

For example, the size range operand:

```
512;1m;4m
```

represents these four size ranges

```
0        to     512 bytes
513      to 1048576 bytes
1048577  to 4194304 bytes
4194305 and greater bytes
```

In this example, a read of size 3 MB would fall in the third size range, a write of size 20 MB would fall in the fourth size range.

A size range operand of = (equal sign) indicates that the current size range is not to be changed. A size range operand of * (asterisk) indicates that the current size range is to be changed to the default size range. A maximum of 15 numbers may be specified, which produces 16 total size ranges.

The default request size ranges are:

```
0        to      255 bytes
256      to      511 bytes
512      to     1023 bytes
1024     to     2047 bytes
2048     to     4095 bytes
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 137

```
4096     to     8191 bytes
8192     to    16383 bytes
16384    to    32767 bytes
32768    to    65535 bytes
65536    to   131071 bytes
131072   to   262143 bytes
262144   to   524287 bytes
524288   to  1048575 bytes
1048576  to  2097151 bytes
2097152  to  4194303 bytes
4194304  and greater bytes
```

The last size range collects all request sizes greater than or equal to 4 MB. The request size ranges can be changed by using the **rhist nr** request.

Specifying the latency ranges for I/O

The latency ranges are used to categorize the I/O according to the latency time, in milliseconds, of the I/O operation. A full set of latency ranges are produced for each size range. The latency ranges are the same for each size range.

The latency ranges are changed using a string of positive decimal numbers separated by semicolons (;). No white space is allowed within the latency range operand. Each number represents the upper bound of the I/O latency time (in milliseconds) for that range. The numbers must be monotonically increasing. If decimal places are present, they are truncated to tenths.

For example, the latency range operand:

```
1.3;4.59;10
```

represents these four latency ranges:

```
 0.0   to   1.3   milliseconds
 1.4   to   4.5   milliseconds
 4.6   to  10.0   milliseconds
10.1  and greater milliseconds
```

In this example, a read that completes in 0.85 milliseconds falls into the first latency range. A write that completes in 4.56 milliseconds falls into the second latency range, due to the truncation.

A latency range operand of = (equal sign) indicates that the current latency range is not to be changed. A latency range operand of * (asterisk) indicates that the current latency range is to be changed to the default latency range. If the latency range operand is missing, * (asterisk) is assumed. A maximum of 15 numbers may be specified, which produces 16 total latency ranges.

The latency times are in milliseconds. The default latency ranges are:

```
0.0     to    1.0   milliseconds
1.1     to   10.0   milliseconds
10.1    to   30.0   milliseconds
30.1    to  100.0   milliseconds
100.1   to  200.0   milliseconds
200.1   to  400.0   milliseconds
400.1   to  800.0   milliseconds
800.1   to 1000.0   milliseconds
1000.1 and greater  milliseconds
```

The last latency range collects all latencies greater than or equal to 1000.1 milliseconds. The latency ranges can be changed by using the **rhist nr** request.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 138

Changing the request histogram facility request size and latency ranges

The **rhist nr** (new range) request allows the user to change the size and latency ranges used in the request histogram facility. The use of **rhist nr** implies an **rhist reset**. Counters for read and write operations are recorded separately. If there are no mounted file systems at the time **rhist nr** is issued, the request still runs. The size range operand appears first, followed by a blank, and then the latency range operand.

Table 15 describes the keywords for the **rhist nr** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 15. Keywords and values for the **mmpmon rhist nr** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `nr`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Processing of rhist nr

Processing of **rhist nr** is as follows:

1. The size range and latency range operands are parsed and checked for validity. If they are not valid, an error is returned and processing terminates.
2. The histogram facility is disabled.
3. The new ranges are created, by defining the following histogram counters:
   a. Two sets, one for read and one for write.
   b. Within each set, one category for each size range.
   c. Within each size range category, one counter for each latency range.

      For example, if the user specifies 11 numbers for the size range operand and 2 numbers for the latency range operand, this produces 12 size ranges, each having 3 latency ranges, because there is one additional range for the top endpoint. The total number of counters is 72: 36 read counters and 36 write counters.
4. The new ranges are made current.
5. The old ranges are discarded. Any accumulated histogram data is lost.

The histogram facility must be explicitly enabled again using **rhist on** to begin collecting histogram data using the new ranges.

The **mmpmon** command does not have the ability to collect data only for read operations, or only for write operations. The **mmpmon** command does not have the ability to specify size or latency ranges that have different values for read and write operations. The **mmpmon** command does not have the ability to specify latency ranges that are unique to a given size range.

Example of mmpmon rhist nr request

Assume that **commandFile** contains this line:

```
rhist nr 512;1m;4m 1.3;4.5;10
```

and this command is issued:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 139

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ nr 512;1m;4m 1.3;4.5;10 _rc_ 0 _t_ 1078929833 _tu_ 765083
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist nr 512;1m;4m 1.3;4.5;10 OK
```

In this case, **mmpmon** has been instructed to keep a total of 32 counters. There are 16 for read and 16 for write. For the reads, there are four size ranges, each of which has four latency ranges. The same is true for the writes. They are as follows:

```
size range            0   to   512    bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
size range          513   to  1048576 bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
size range       1048577  to  4194304 bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
size range       4194305 and greater bytes
  latency range       0.0   to   1.3    milliseconds
  latency range       1.4   to   4.5    milliseconds
  latency range       4.6   to  10.0    milliseconds
  latency range       10.1 and greater  milliseconds
```

In this example, a read of size 15 MB that completes in 17.8 milliseconds would fall in the last latency range listed here. When this read completes, the counter for the last latency range will be increased by one.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

An example of an unsuccessful response is:

```
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ nr 512;1m;4m 1;4;8;2 _rc_ 22 _t_ 1078929596 _tu_ 161683
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist nr 512;1m;4m 1;4;8;2 status 22 range error
```

In this case, the last value in the latency range, 2, is out of numerical order.

Note that the request **rhist nr = =** does not make any changes. It is ignored.

Disabling the request histogram facility

The **rhist off** request disables the request histogram facility. The data objects remain persistent, and the data they contain is not disturbed. This data is not updated again until **rhist on** is issued. **rhist off** may be combined with **rhist on** as often as desired. If there are no mounted file systems at the time **rhist off** is issued, the facility is still disabled. The response is a single string.

Table 16 on page 119 describes the keywords for the **rhist off** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 140

Table 16. Keywords and values for the **mmpmon rhist off** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `off`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Example of mmpmon rhist off request

Assume that **commandFile** contains this line:

```
rhist off
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ off _rc_ 0 _t_ 1066938820 _tu_ 5755
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist off OK
```

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

```
mmpmon node 199.18.1.8 name node1 rhist off status 16
lock is busy
```

Enabling the request histogram facility

The **rhist on** request enables the request histogram facility. When invoked the first time, this request creates the necessary data objects to support histogram data gathering. This request may be combined with **rhist off** (or another **rhist on**) as often as desired. If there are no mounted file systems at the time **rhist on** is issued, the facility is still enabled. The response is a single string.

Table 17 describes the keywords for the **rhist on** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 17. Keywords and values for the **mmpmon rhist on** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `on`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 141

Example of mmpmon rhist on request

Assume that **commandFile** contains this line:

```
rhist on
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ on _rc_ 0 _t_ 1066936484 _tu_ 179346
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist on OK
```

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

```
mmpmon node 199.18.1.8 name node1 rhist on status 16
lock is busy
```

Displaying the request histogram facility pattern

The **rhist p** request returns the entire enumeration of the request size and latency ranges. The facility must be enabled for a pattern to be returned. If there are no mounted file systems at the time this request is issued, the request still runs and returns data. The pattern is displayed for both read and write.

Table 18 describes the keywords for the **rhist p** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 18. Keywords and values for the **mmpmon rhist p** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `p`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.
_k_	The kind, **r** or **w**, (read or write) depending on what the statistics are for.
_R_	Request size range, minimum and maximum number of bytes.
_L_	Latency range, minimum and maximum, in milliseconds.

The request size ranges are in bytes. The zero value used for the upper limit of the last size range means 'and above'. The request size ranges can be changed by using the **rhist nr** request.

The latency times are in milliseconds The zero value used for the upper limit of the last latency range means 'and above'. The latency ranges can be changed by using the **rhist nr** request.

The **rhist p** request allows an application to query for the entire latency pattern. The application can then configure itself accordingly. Since latency statistics are reported only for ranges with nonzero counts, the statistics responses may be sparse. By querying for the pattern, an application can be certain to learn the complete histogram set. The user may have changed the pattern using the **rhist nr** request. For this reason, an application should query for the pattern and analyze it before requesting statistics.

If the facility has never been enabled, the **_rc_** field will be nonzero. An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 142

If the facility has been previously enabled, the **rhist p** request will still display the pattern even if **rhist off** is currently in effect.

If there are no mounted file systems at the time **rhist p** is issued, the pattern is still displayed.

Example of mmpmon rhist p request

Assume that **commandFile** contains this line:

```
rhist p
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The response contains all the latency ranges inside each of the request ranges. The data are separate for read and write:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ r
... data for reads ...
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ w
... data for writes ...
_end_
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist p OK read
... data for reads ...
mmpmon node 199.188.1.8 name node1 rhist p OK write
... data for writes ...
```

Here is an example of data for reads:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ r
_R_          0            255
_L_        0.0            1.0
_L_        1.1           10.0
_L_       10.1           30.0
_L_       30.1          100.0
_L_      100.1          200.0
_L_      200.1          400.0
_L_      400.1          800.0
_L_      800.1         1000.0
_L_     1000.1              0
_R_        256            511
_L_        0.0            1.0
_L_        1.1           10.0
_L_       10.1           30.0
_L_       30.1          100.0
_L_      100.1          200.0
_L_      200.1          400.0
_L_      400.1          800.0
_L_      800.1         1000.0
_L_     1000.1              0
_R_        512           1023
_L_        0.0            1.0
_L_        1.1           10.0
_L_       10.1           30.0
_L_       30.1          100.0
_L_      100.1          200.0
_L_      200.1          400.0
_L_      400.1          800.0
_L_      800.1         1000.0
_L_     1000.1              0
...
_R_    4194304              0
_L_        0.0            1.0
_L_        1.1           10.0
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 143

```
_L_      10.1           30.0
_L_      30.1          100.0
_L_     100.1          200.0
_L_     200.1          400.0
_L_     400.1          800.0
_L_     800.1         1000.0
_L_    1000.1              0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist p OK read
size range         0 to          255
  latency range    0.0 to          1.0
  latency range    1.1 to         10.0
  latency range   10.1 to         30.0
  latency range   30.1 to        100.0
  latency range  100.1 to        200.0
  latency range  200.1 to        400.0
  latency range  400.1 to        800.0
  latency range  800.1 to       1000.0
  latency range 1000.1 to            0
size range       256 to          511
  latency range    0.0 to          1.0
  latency range    1.1 to         10.0
  latency range   10.1 to         30.0
  latency range   30.1 to        100.0
  latency range  100.1 to        200.0
  latency range  200.1 to        400.0
  latency range  400.1 to        800.0
  latency range  800.1 to       1000.0
  latency range 1000.1 to            0
size range       512 to         1023
  latency range    0.0 to          1.0
  latency range    1.1 to         10.0
  latency range   10.1 to         30.0
  latency range   30.1 to        100.0
  latency range  100.1 to        200.0
  latency range  200.1 to        400.0
  latency range  400.1 to        800.0
  latency range  800.1 to       1000.0
  latency range 1000.1 to            0
...
size range   4194304 to           0
  latency range    0.0 to          1.0
  latency range    1.1 to         10.0
  latency range   10.1 to         30.0
  latency range   30.1 to        100.0
  latency range  100.1 to        200.0
  latency range  200.1 to        400.0
  latency range  400.1 to        800.0
  latency range  800.1 to       1000.0
  latency range 1000.1 to            0
```

If the facility has never been enabled, the **_rc_** field will be nonzero.

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ p _rc_ 1 _t_ 1066939007 _tu_ 386241
```

If the **-p** flag is not specified, the output is similar to this:

```
mmpmon node 199.18.1.8 name node1 rhist p status 1
not yet enabled
```

Resetting the request histogram facility data to zero

The **rhist reset** request resets the histogram statistics. Table 19 on page 123 describes the keywords for the **rhist reset** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag. The response is a single string.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 144

Table 19. Keywords and values for the **mmpmon rhist reset** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `reset`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

If the facility has been previously enabled, the reset request will still reset the statistics even if **rhist off** is currently in effect. If there are no mounted file systems at the time **rhist reset** is issued, the statistics are still reset.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Example of mmpmon rhist reset request

Assume that **commandFile** contains this line:

```
rhist reset
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 0 _t_ 1066939007 _tu_ 386241
```

If the **-p** flag is not specified, the output is similar to:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 0 _t_ 1066939007 _tu_ 386241
```

If the facility has never been enabled, the **_rc_** value will be nonzero:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 1 _t_ 1066939143 _tu_ 148443
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist reset status 1
not yet enabled
```

Displaying the request histogram facility statistics values

The **rhist s** request returns the current values for all latency ranges which have a nonzero count.

Table 20 describes the keywords for the **rhist s** response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Table 20. Keywords and values for the **mmpmon rhist s** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
req	The action requested. In this case, the value is `s`.
_rc_	Indicates the status of the operation.
_t_	Indicates the current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	Microseconds part of the current time of day.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 145

Table 20. Keywords and values for the **mmpmon rhist s** *response (continued)*

Keyword	Description
_k_	The kind, **r** or **w**, (read or write) depending on what the statistics are for.
_R_	Request size range, minimum and maximum number of bytes.
_NR_	Number of requests that fell in this size range.
_L_	Latency range, minimum and maximum, in milliseconds.
_NL_	Number of requests that fell in this latency range. The sum of all **_NL_** values for a request size range equals the **_NR_** value for that size range.

If the facility has been previously enabled, the **rhist s** request will still display the statistics even if **rhist off** is currently in effect. This allows turning the histogram statistics on and off between known points and reading them later. If there are no mounted file systems at the time **rhist s** is issued, the statistics are still displayed.

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Example of mmpmon rhist s request

Assume that **commandFile** contains this line:

```
rhist s
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ s _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ r
_R_       65536      131071 _NR_       32640
_L_         0.0         1.0 _NL_       25684
_L_         1.1        10.0 _NL_        4826
_L_        10.1        30.0 _NL_        1666
_L_        30.1       100.0 _NL_         464
_R_      262144      524287 _NR_        8160
_L_         0.0         1.0 _NL_        5218
_L_         1.1        10.0 _NL_         871
_L_        10.1        30.0 _NL_        1863
_L_        30.1       100.0 _NL_         208
_R_     1048576     2097151 _NR_        2040
_L_         1.1        10.0 _NL_         558
_L_        10.1        30.0 _NL_         809
_L_        30.1       100.0 _NL_         673
_rhist_ _n_ 199.18.2.5 _nn_ node1 _req_ s _rc_ 0 _t_ 1066939007 _tu_ 386241 _k_ w
_R_      131072      262143 _NR_       12240
_L_         0.0         1.0 _NL_       10022
_L_         1.1        10.0 _NL_        1227
_L_        10.1        30.0 _NL_         783
_L_        30.1       100.0 _NL_         208
_R_      262144      524287 _NR_        6120
_L_         0.0         1.0 _NL_        4419
_L_         1.1        10.0 _NL_         791
_L_        10.1        30.0 _NL_         733
_L_        30.1       100.0 _NL_         177
_R_      524288     1048575 _NR_        3060
_L_         0.0         1.0 _NL_        1589
_L_         1.1        10.0 _NL_         581
_L_        10.1        30.0 _NL_         664
_L_        30.1       100.0 _NL_         226
_R_     2097152     4194303 _NR_         762
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 146

```
_L_        1.1         2.0 _NL_        203
_L_       10.1        30.0 _NL_        393
_L_       30.1       100.0 _NL_        166
_end_
```

This small example shows that the reports for read and write may not present the same number of ranges or even the same ranges. Only those ranges with nonzero counters are represented in the response. This is true for both the request size ranges and the latency ranges within each request size range.

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.2.5 name node1 rhist s OK timestamp 1066933849/93804 read
size range            65536 to    131071 count      32640
  latency range         0.0 to       1.0 count      25684
  latency range         1.1 to      10.0 count       4826
  latency range        10.1 to      30.0 count       1666
  latency range        30.1 to     100.0 count        464
size range           262144 to    524287 count       8160
  latency range         0.0 to       1.0 count       5218
  latency range         1.1 to      10.0 count        871
  latency range        10.1 to      30.0 count       1863
  latency range        30.1 to     100.0 count        208
size range          1048576 to    2097151 count      2040
  latency range         1.1 to       10.0 count       558
  latency range        10.1 to       30.0 count       809
  latency range        30.1 to      100.0 count       673
mmpmon node 199.18.2.5 name node1 rhist s OK timestamp 1066933849/93968 write
size range           131072 to     262143 count     12240
  latency range         0.0 to        1.0 count     10022
  latency range         1.1 to       10.0 count      1227
  latency range        10.1 to       30.0 count       783
  latency range        30.1 to      100.0 count       208
size range           262144 to     524287 count      6120
  latency range         0.0 to        1.0 count      4419
  latency range         1.1 to       10.0 count       791
  latency range        10.1 to       30.0 count       733
  latency range        30.1 to      100.0 count       177
size range           524288 to    1048575 count      3060
  latency range         0.0 to        1.0 count      1589
  latency range         1.1 to       10.0 count       581
  latency range        10.1 to       30.0 count       664
  latency range        30.1 to      100.0 count       226
size range          2097152 to    4194303 count       762
  latency range         1.1 to         2.0 count      203
  latency range        10.1 to        30.0 count      393
  latency range        30.1 to       100.0 count     166
```

If the facility has never been enabled, the **_rc_** value will be nonzero:

```
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ reset _rc_ 1 _t_ 1066939143 _tu_ 148443
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 rhist reset status 1
not yet enabled
```

An **_rc_** value of 16 indicates that the histogram operations lock is busy. Retry the request.

Displaying mmpmon version

The **ver** request returns a string containing version information. Table 21 on page 126 describes the keywords for the **ver** (version) response, in the order that they appear in the output. These keywords are used only when **mmpmon** is invoked with the **-p** flag.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 147

Table 21. Keywords and values for the **mmpmon ver** *response*

Keyword	Description
_n_	IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The hostname that corresponds to the IP address (the **_n_** value).
_v_	The version of **mmpmon**.
_lv_	The level of **mmpmon**.
_vt_	The fix level variant of **mmpmon**.

Example of mmpmon ver request

Assume that **commandFile** contains this line:

```
ver
```

and this command is issued:

```
mmpmon -p -i commandFile
```

The output is similar to this:

```
_ver_ _n_ 199.18.1.8 _nn_ node1 _v_ 3 _lv_ 3 _vt_ 0
```

If the **-p** flag is not specified, the output is similar to:

```
mmpmon node 199.18.1.8 name node1 version 3.3.0
```

Example mmpmon scenarios and how to analyze and interpret their results

This topic is an illustration of how **mmpmon** is used to analyze I/O data and draw conclusions based on it.

The **fs_io_s** and **io_s** requests are used to determine a number of GPFS I/O parameters and their implication for overall performance. The **rhist** requests are used to produce histogram data about I/O sizes and latency times for I/O requests. The request *source* and prefix directive *once* allow the user of **mmpmon** to more finely tune its operation.

fs_io_s and io_s output - how to aggregate and analyze the results

The output from the **fs_io_s** and **io_s** requests can be used to determine:

1. The I/O service rate of a node, from the application point of view. The **io_s** request presents this as a sum for the entire node, while **fs_io_s** presents the data per file system. A rate can be approximated by taking the **_br_** (bytes read) or **_bw_** (bytes written) values from two successive invocations of **fs_io_s** (or **io_s_**) and dividing by the difference of the sums of the individual **_t_** and **_tu_** values (seconds and microseconds).

   This must be done for a number of samples, with a reasonably small time between samples, in order to get a rate which is reasonably accurate. Since we are sampling the information at a given interval, inaccuracy can exist if the I/O load is not smooth over the sampling time.

   For example, here is a set of samples taken approximately one second apart, when it was known that continuous I/O activity was occurring:

   ```
   _fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862476 _tu_ 634939 _cl_ cluster1.xxx.com
   _fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 3737124864 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 3570 _dir_ 0 _iu_ 5
   _fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862477 _tu_ 645988 _cl_ cluster1.xxx.com
   _fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 3869245440 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 3696 _dir_ 0 _iu_ 5
   _fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862478 _tu_ 647477 _cl_ cluster1.xxx.com
   _fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4120903680 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 3936 _dir_ 0 _iu_ 5
   ```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 148

```
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862479 _tu_ 649363 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4309647360 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4116 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862480 _tu_ 650795 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4542431232 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4338 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862481 _tu_ 652515 _cl_ cluster1.ibm.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4743757824 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4530 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862482 _tu_ 654025 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 4963958784 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4740 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862483 _tu_ 655782 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 5177868288 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 4944 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862484 _tu_ 657523 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 5391777792 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 5148 _dir_ 0 _iu_ 5
_fs_io_s_ _n_ 199.18.1.3 _nn_ node1 _rc_ 0 _t_ 1095862485 _tu_ 665909 _cl_ cluster1.xxx.com
_fs_ gpfs1m _d_ 3 _br_ 0 _bw_ 5599395840 _oc_ 4 _cc_ 3 _rdc_ 0 _wc_ 5346 _dir_ 0 _iu_ 5
```

This simple **awk** script performs a basic rate calculation:

```
BEGIN {
  count=0;
  prior_t=0;
  prior_tu=0;
  prior_br=0;
  prior_bw=0;
}

{
  count++;

  t = $9;
  tu = $11;
  br = $19;
  bw = $21;

  if(count > 1)
  {
    delta_t = t-prior_t;
    delta_tu = tu-prior_tu;
    delta_br = br-prior_br;
    delta_bw = bw-prior_bw;
    dt = delta_t + (delta_tu / 1000000.0);
    if(dt > 0) {
      rrate = (delta_br / dt) / 1000000.0;
      wrate = (delta_bw / dt) / 1000000.0;

printf("%5.1f MB/sec read %5.1f MB/sec write\n",rrate,wrate);
    }
  }

  prior_t=t;
  prior_tu=tu;
  prior_br=br;
  prior_bw=bw;
}
```

The calculated service rates for each adjacent pair of samples is:

```
  0.0 MB/sec read    130.7 MB/sec write
  0.0 MB/sec read    251.3 MB/sec write
  0.0 MB/sec read    188.4 MB/sec write
  0.0 MB/sec read    232.5 MB/sec write
  0.0 MB/sec read    201.0 MB/sec write
  0.0 MB/sec read    219.9 MB/sec write
  0.0 MB/sec read    213.5 MB/sec write
  0.0 MB/sec read    213.5 MB/sec write
  0.0 MB/sec read    205.9 MB/sec write
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 149

Since these are discrete samples, there can be variations in the individual results. For example, there may be other activity on the node or interconnection fabric. I/O size, file system block size, and buffering also affect results. There can be many reasons why adjacent values differ. This must be taken into account when building analysis tools that read **mmpmon** output and interpreting results.

For example, suppose a file is read for the first time and gives results like this.

```
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
 92.1 MB/sec read      0.0 MB/sec write
 89.0 MB/sec read      0.0 MB/sec write
 92.1 MB/sec read      0.0 MB/sec write
 90.0 MB/sec read      0.0 MB/sec write
 96.3 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
```

If most or all of the file remains in the GPFS cache, the second read may give quite different rates:

```
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
235.5 MB/sec read      0.0 MB/sec write
287.8 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
  0.0 MB/sec read      0.0 MB/sec write
```

Considerations such as these need to be taken into account when looking at application I/O service rates calculated from sampling **mmpmon** data.

2. Usage patterns, by sampling at set times of the day (perhaps every half hour) and noticing when the largest changes in I/O volume occur. This does not necessarily give a rate (since there are too few samples) but it can be used to detect peak usage periods.
3. If some nodes service significantly more I/O volume than others over a given time span.
4. When a parallel application is split across several nodes, and is the only significant activity in the nodes, how well the I/O activity of the application is distributed.
5. The total I/O demand that applications are placing on the cluster. This is done by obtaining results from **fs_io_s** and **io_s** in aggregate for all nodes in a cluster.
6. The rate data may appear to be erratic. Consider this example:

```
 0.0 MB/sec read    0.0 MB/sec write
 6.1 MB/sec read    0.0 MB/sec write
92.1 MB/sec read    0.0 MB/sec write
89.0 MB/sec read    0.0 MB/sec write
12.6 MB/sec read    0.0 MB/sec write
 0.0 MB/sec read    0.0 MB/sec write
 0.0 MB/sec read    0.0 MB/sec write
 8.9 MB/sec read    0.0 MB/sec write
92.1 MB/sec read    0.0 MB/sec write
90.0 MB/sec read    0.0 MB/sec write
96.3 MB/sec read    0.0 MB/sec write
 4.8 MB/sec read    0.0 MB/sec write
 0.0 MB/sec read    0.0 MB/sec write
```

The low rates which appear before and after each group of higher rates can be due to the I/O requests occurring late (in the leading sampling period) and ending early (in the trailing sampling period.) This gives an apparently low rate for those sampling periods.

The zero rates in the middle of the example could be caused by reasons such as no I/O requests reaching GPFS during that time period (the application issued none, or requests were satisfied by buffered data at a layer above GPFS), the node becoming busy with other work (causing the application to be undispatched), or other reasons.

Request histogram (rhist) output - how to aggregate and analyze the results

The output from the **rhist** requests can be used to determine:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 150

1. The number of I/O requests in a given size range. The sizes may vary based on operating system, explicit application buffering, and other considerations. This information can be used to help determine how well an application or set of applications is buffering its I/O. For example, if are there many very small or many very large I/O transactions. A large number of overly small or overly large I/O requests may not perform as well as an equivalent number of requests whose size is tuned to the file system or operating system parameters.
2. The number of I/O requests in a size range that have a given latency time. Many factors can affect the latency time, including but not limited to: system load, interconnection fabric load, file system block size, disk block size, disk hardware characteristics, and the operating system on which the I/O request is issued.

Using request *source* and prefix directive *once*

The **source** request causes **mmpmon** to read requests from a file, and when finished return to reading requests from the input stream.

The prefix directive **once** can be placed in front of any **mmpmon** request. The **once** prefix indicates that the request be run only once, irrespective of the setting of the **-r** flag on the **mmpmon** command. It is useful for requests that do not need to be issued more than once, such as to set up the node list or turn on the request histogram facility.

These rules apply when using the **once** prefix directive and **source** request:

1. **once** with nothing after it is an error that terminates **mmpmon** processing.
2. A file invoked with the **source** request may contain **source** requests, causing file nesting of arbitrary depth. No check is done for loops in this situation.
3. The request **once source** *filename* causes the **once** prefix to be applied to all the **mmpmon** requests in *filename*, including any **source** requests in the file.
4. If a *filename* specified with the **source** request cannot be opened for read, an error is returned and **mmpmon** terminates.
5. If the **-r** flag on the **mmpmon** command has any value other than one, and all requests are prefixed with **once**, **mmpmon** runs all the requests once, issues a message, and then terminates.

An example of *once* and *source* usage

This example illustrates the use of **once** and **source**. This command is issued:

```
mmpmon -p -i command.file -r 0 -d 5000 | tee output.file
```

File **command.file** consists of this:

```
once source mmpmon.header
once rhist nr 512;1024;2048;4096 =
once rhist on
source mmpmon.commands
```

File **mmpmon.header** consists of this:

```
ver
reset
```

File **mmpmon.commands** consists of this:

```
fs_io_s
rhist s
```

The **output.file** is similar to this:

```
_ver_ _n_ 199.18.1.8 _nn_ node1 _v_ 2 _lv_ 4 _vt_ 0
_reset_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770129 _tu_ 511981
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ nr 512;1024;2048;4096 = _rc_ 0 _t_ 1129770131 _tu_ 524674
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ on _rc_ 0 _t_ 1129770131 _tu_ 524921
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770131 _tu_ 525062 _cl_ node1.localdomain
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 151

```
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770131 _tu_ 525062 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770131 _tu_ 525220 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770131 _tu_ 525228 _k_ w
_end_
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770136 _tu_ 526685 _cl_ node1.localdomain
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770136 _tu_ 526685 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 395018 _oc_ 504 _cc_ 252 _rdc_ 0 _wc_ 251 _dir_ 0 _iu_ 147
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770136 _tu_ 526888 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770136 _tu_ 526896 _k_ w
_R_ 0 512 _NR_ 169
_L_ 0.0 1.0 _NL_ 155
_L_ 1.1 10.0 _NL_ 7
_L_ 10.1 30.0 _NL_ 1
_L_ 30.1 100.0 _NL_ 4
_L_ 100.1 200.0 _NL_ 2
_R_ 513 1024 _NR_ 16
_L_ 0.0 1.0 _NL_ 15
_L_ 1.1 10.0 _NL_ 1
_R_ 1025 2048 _NR_ 3
_L_ 0.0 1.0 _NL_ 32
_R_ 2049 4096 _NR_ 18
_L_ 0.0 1.0 _NL_ 18
_R_ 4097 0 _NR_ 16
_L_ 0.0 1.0 _NL_ 16
_end_
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770141 _tu_ 528613 _cl_ node1.localdomain
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 0
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770141 _tu_ 528613 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 823282 _oc_ 952 _cc_ 476 _rdc_ 0 _wc_ 474 _dir_ 0 _iu_ 459
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770141 _tu_ 528812 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770141 _tu_ 528820 _k_ w
_R_ 0 512 _NR_ 255
_L_ 0.0 1.0 _NL_ 241
_L_ 1.1 10.0 _NL_ 7
_L_ 10.1 30.0 _NL_ 1
_L_ 30.1 100.0 _NL_ 4
_L_ 100.1 200.0 _NL_ 2
_R_ 513 1024 _NR_ 36
_L_ 0.0 1.0 _NL_ 35
_L_ 1.1 10.0 _NL_ 1
_R_ 1025 2048 _NR_ 90
_L_ 0.0 1.0 _NL_ 90
_R_ 2049 4096 _NR_ 55
_L_ 0.0 1.0 _NL_ 55
_R_ 4097 0 _NR_ 38
_L_ 0.0 1.0 _NL_ 37
_L_ 1.1 10.0 _NL_ 1
_end_
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770146 _tu_ 530570 _cl_ node1.localdomain
_fs_ gpfs1 _d_ 1 _br_ 0 _bw_ 0 _oc_ 0 _cc_ 0 _rdc_ 0 _wc_ 0 _dir_ 0 _iu_ 1
_fs_io_s_ _n_ 199.18.1.8 _nn_ node1 _rc_ 0 _t_ 1129770146 _tu_ 530570 _cl_ node1.localdomain
_fs_ gpfs2 _d_ 2 _br_ 0 _bw_ 3069915 _oc_ 1830 _cc_ 914 _rdc_ 0 _wc_ 901 _dir_ 0 _iu_ 1070
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770146 _tu_ 530769 _k_ r
_rhist_ _n_ 199.18.1.8 _nn_ node1 _req_ s _rc_ 0 _t_ 1129770146 _tu_ 530778 _k_ w
_R_ 0 512 _NR_ 526
_L_ 0.0 1.0 _NL_ 501
_L_ 1.1 10.0 _NL_ 14
_L_ 10.1 30.0 _NL_ 2
_L_ 30.1 100.0 _NL_ 6
_L_ 100.1 200.0 _NL_ 3
_R_ 513 1024 _NR_ 74
_L_ 0.0 1.0 _NL_ 70
_L_ 1.1 10.0 _NL_ 4
_R_ 1025 2048 _NR_ 123
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 152

```
_L_ 0.0 1.0 _NL_ 117
_L_ 1.1 10.0 _NL_ 6
_R_ 2049 4096 _NR_ 91
_L_ 0.0 1.0 _NL_ 84
_L_ 1.1 10.0 _NL_ 7
_R_ 4097 0 _NR_ 87
_L_ 0.0 1.0 _NL_ 81
_L_ 1.1 10.0 _NL_ 6
_end_
.............. and so forth ......................
```

If this command is issued with the same file contents:

```
mmpmon -i command.file -r 0 -d 5000 | tee output.file.english
```

The file **output.file.english** is similar to this:

```
mmpmon node 199.18.1.8 name node1 version 3.1.0
mmpmon node 199.18.1.8 name node1 reset OK
mmpmon node 199.18.1.8 name node1 rhist nr 512;1024;2048;4096 = OK
mmpmon node 199.18.1.8 name node1 rhist on OK
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                  1
timestamp:      1129770175/950895
bytes read:             0
bytes written:          0
opens:                  0
closes:                 0
reads:                  0
writes:                 0
readdir:                0
inode updates:          0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                  2
timestamp:      1129770175/950895
bytes read:             0
bytes written:

opens:                  0
closes:                 0
reads:                  0
writes:                 0
readdir:                0
inode updates:          0
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770175/951117
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770175/951125
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                  1
timestamp:      1129770180/952462
bytes read:             0
bytes written:          0
opens:                  0
closes:                 0
reads:                  0
writes:                 0
readdir:                0
inode updates:          0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 153

```
disks:                   2
timestamp:      1129770180/952462
bytes read:              0
bytes written:      491310
opens:                659
closes:               329
reads:                  0
writes:               327
readdir:                0
inode updates:         74
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770180/952711
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770180/952720
size range               0 to        512 count        214
  latency range        0.0 to        1.0 count        187
  latency range        1.1 to       10.0 count         15
  latency range       10.1 to       30.0 count          6
  latency range       30.1 to      100.0 count          5
  latency range      100.1 to      200.0 count          1
size range             513 to       1024 count         27
  latency range        0.0 to        1.0 count         26
  latency range      100.1 to      200.0 count          1
size range            1025 to       2048 count         32
  latency range        0.0 to        1.0 count         29
  latency range        1.1 to       10.0 count          1
  latency range       30.1 to      100.0 count          2
size range            2049 to       4096 count         31
  latency range        0.0 to        1.0 count         30
  latency range       30.1 to      100.0 count          1
size range            4097 to          0 count         23
  latency range        0.0 to        1.0 count         23
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                   1
timestamp:      1129770185/954401
bytes read:              0
bytes written:           0
opens:                   0
closes:                  0
reads:                   0
writes:                  0
readdir:                 0
inode updates:           0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                   2
timestamp:      1129770185/954401
bytes read:              0
bytes written:     1641935
opens:                1062
closes:                531
reads:                   0
writes:                529
readdir:                 0
inode updates:         523
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770185/954658
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770185/954667
size range               0 to        512 count        305
  latency range        0.0 to        1.0 count        270
  latency range        1.1 to       10.0 count         21
  latency range       10.1 to       30.0 count          6
  latency range       30.1 to      100.0 count          6
  latency range      100.1 to      200.0 count          2
size range             513 to       1024 count         39
  latency range        0.0 to        1.0 count         36
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 154

```
  latency range        1.1 to       10.0 count          1
  latency range       30.1 to      100.0 count          1
  latency range      100.1 to      200.0 count          1
size range            1025 to       2048 count         89
  latency range        0.0 to        1.0 count         84
  latency range        1.1 to       10.0 count          2
  latency range       30.1 to      100.0 count          3
size range            2049 to       4096 count         56
  latency range        0.0 to        1.0 count         54
  latency range        1.1 to       10.0 count          1
  latency range       30.1 to      100.0 count          1
size range            4097 to          0 count         40
  latency range        0.0 to        1.0 count         39
  latency range        1.1 to       10.0 count          1
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                 1
timestamp:      1129770190/956480
bytes read:            0
bytes written:         0
opens:                 0
closes:                0
reads:                 0
writes:                0
readdir:               0
inode updates:         0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                 2
timestamp:      1129770190/956480
bytes read:            0
bytes written:     3357414
opens:              1940
closes:              969
reads:                 0
writes:              952
readdir:               0
inode updates:      1101
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770190/956723
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770190/956732
size range             0 to        512 count        539
  latency range        0.0 to        1.0 count        494
  latency range        1.1 to       10.0 count         29
  latency range       10.1 to       30.0 count          6
  latency range       30.1 to      100.0 count          8
  latency range      100.1 to      200.0 count          2
size range             513 to       1024 count         85
  latency range        0.0 to        1.0 count         81
  latency range        1.1 to       10.0 count          2
  latency range       30.1 to      100.0 count          1
  latency range      100.1 to      200.0 count          1
size range            1025 to       2048 count        133
  latency range        0.0 to        1.0 count        124
  latency range        1.1 to       10.0 count          5
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          3
size range            2049 to       4096 count         99
  latency range        0.0 to        1.0 count         91
  latency range        1.1 to       10.0 count          6
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          1
size range            4097 to          0 count         95
  latency range        0.0 to        1.0 count         90
  latency range        1.1 to       10.0 count          4
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 155

```
  latency range       10.1 to       30.0 count          1
mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs1
disks:                  1
timestamp:      1129770195/958310
bytes read:             0
bytes written:          0
opens:                  0
closes:                 0
reads:                  0
writes:                 0
readdir:                0
inode updates:          0

mmpmon node 199.18.1.8 name node1 fs_io_s OK
cluster:        node1.localdomain
filesystem:     gpfs2
disks:                  2
timestamp:      1129770195/958310
bytes read:             0
bytes written:     3428107
opens:                2046
closes:               1023
reads:                   0
writes:                997
readdir:                 0
inode updates:        1321
mmpmon node 199.18.1.8 name node1 rhist s OK read timestamp 1129770195/958568
mmpmon node 199.18.1.8 name node1 rhist s OK write timestamp 1129770195/958577
size range               0 to        512 count        555
  latency range        0.0 to        1.0 count        509
  latency range        1.1 to       10.0 count         30
  latency range       10.1 to       30.0 count          6
  latency range       30.1 to      100.0 count          8
  latency range      100.1 to      200.0 count          2
size range             513 to       1024 count         96
  latency range        0.0 to        1.0 count         92
  latency range        1.1 to       10.0 count          2
  latency range       30.1 to      100.0 count          1
  latency range      100.1 to      200.0 count          1
size range            1025 to       2048 count        143
  latency range        0.0 to        1.0 count        134
  latency range        1.1 to       10.0 count          5
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          3
size range            2049 to       4096 count        103
  latency range        0.0 to        1.0 count         95
  latency range        1.1 to       10.0 count          6
  latency range       10.1 to       30.0 count          1
  latency range       30.1 to      100.0 count          1
size range            4097 to          0 count        100
  latency range        0.0 to        1.0 count         95
latency range        1.1 to         10.0 count          4
latency range       10.1 to         30.0 count          1
.............. and so forth ......................
```

Other information about mmpmon output

When interpreting the results from the **mmpmon** output there are several points to consider.

Consider these important points:

- On a node acting as a server of a GPFS file system to NFS clients, NFS I/O is accounted for in the statistics. However, the I/O is that which goes between GPFS and NFS. If NFS caches data, in order to achieve better performance, this activity is not recorded.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 156

- I/O requests made at the application level may not be exactly what is reflected to GPFS. This is dependent on the operating system, and other factors. For example, an application read of 100 bytes may result in obtaining, and caching, a 1 MB block of data at a code level above GPFS (such as the libc I/O layer.) . Subsequent reads within this block result in no additional requests to GPFS.
- The counters kept by **mmpmon** are not atomic and may not be exact in cases of high parallelism or heavy system load. This design minimizes the performance impact associated with gathering statistical data.
- Reads from data cached by GPFS will be reflected in statistics and histogram data. Reads and writes to data cached in software layers above GPFS will be reflected in statistics and histogram data when those layers actually call GPFS for I/O.
- Activity from snapshots affects statistics. I/O activity necessary to maintain a snapshot is counted in the file system statistics.
- Some (generally minor) amount of activity in the root directory of a file system is reflected in the statistics of the file system manager node, and not the node which is running the activity.
- The open count also includes **creat()** call counts.

Counter sizes and counter wrapping

The **mmpmon** command may be run continuously for extended periods of time. The user must be aware that counters may wrap. This information applies to the counters involved:

- The statistical counters used for the **io_s** and **fs_io_s** requests are maintained by GPFS at all times, even when **mmpmon** has not been invoked. It is suggested that you use the **reset** request prior to starting a sequence of **io_s** or **fs_io_s** requests.
- The bytes read and bytes written counters are unsigned 64-bit integers. They are used in the **fs_io_s** and **io_s** requests, as the **_br_** and **_bw_** fields.
- The counters associated with the **rhist** requests are updated only when the request histogram facility has been enabled.
- The counters used in the **rhist** requests are unsigned 64-bit integers.
- All other counters are unsigned 32-bit integers.

Return codes from mmpmon

These are the return codes that can appear in the **_rc_** field:

0 Successful completion.

1 One of these has occurred:

1. For the **fs_io_s** request, no file systems are mounted.
2. For an **rhist** request, a request was issued that requires the request histogram facility to be enabled, but it is not. The facility is not enabled if:
   - Since the last **mmstartup** was issued, **rhist on** was never issued.
   - **rhist nr** was issued and **rhist on** was not issued afterwards.

2 For one of the **nlist** requests, the node name is not recognized.

13 For one of the **nlist** requests, the node name is a remote node, which is not allowed.

16 For one of the **rhist** requests, the histogram operations lock is busy. Retry the request.

17 For one of the **nlist** requests, the node name is already in the node list.

22 For one of the **rhist** requests, the size or latency range parameters were not in ascending order or were otherwise incorrect.

233 For one of the **nlist** requests, the specified node is not joined to the cluster.

668 For one of the **nlist** requests, quorum has been lost in the cluster.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 157

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 158

Chapter 10. GPFS SNMP support

GPFS supports the use of the SNMP protocol for monitoring the status and configuration of the GPFS cluster. Using an SNMP application, the system administrator can get a detailed view of the system and be instantly notified of important events, such as a node or disk failure.

The Simple Network Management Protocol (SNMP) is an application-layer protocol that facilitates the exchange of management information between network devices. It is part of the Transmission Control Protocol/Internet Protocol (TCP/IP) protocol suite. SNMP enables network administrators to manage network performance, find and solve network problems, and plan for network growth.

SNMP consists of commands to enumerate, read, and write managed variables that are defined for a particular device. It also has a **trap** command, for communicating events asynchronously.

The variables are organized as instances of objects, known as management information bases (MIBs). MIBs are organized in a hierarchical tree by organization (for example, IBM). A GPFS MIB is defined for monitoring many aspects of GPFS.

An SNMP agent software architecture typically consists of a master agent and a set of subagents, which communicate with the master agent through a specific agent/subagent protocol (the AgentX protocol in this case). Each subagent handles a particular system or type of device. A GPFS SNMP subagent is provided, which maps the SNMP objects and their values.

Installing Net-SNMP

The SNMP subagent runs on the collector node of the GPFS cluster. The collector node is designated by the system administrator.

See “Collector node administration” on page 139 for information.

The Net-SNMP master agent (also called the SNMP daemon, or **snmpd**) must be installed on the collector node to communicate with the GPFS subagent and with your SNMP management application. Net-SNMP is included in most Linux distributions and should be supported by your Linux vendor. Source and binaries for several platforms are available at this Web site:

```
http://www.net-snmp.org/download.html
```

Note: Currently, the collector node must run on the Linux operating system. For an up-to-date list of supported operating systems, specific distributions, and other dependencies, refer to the GPFS FAQ in the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) or GPFS FAQ in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfsclustersfaq.html).

The GPFS subagent expects to find the following libraries:

```
libnetsnmpagent.so      -- from Net-SNMP
libnetsnmphelpers.so    -- from Net-SNMP
libnetsnmpmibs.so       -- from Net-SNMP
libnetsnmp.so           -- from Net-SNMP
libwrap.so              -- from TCP Wrappers
libcrypto.so            -- from OpenSSL
```

Note: TCP Wrappers and OpenSSL should have been prerequisited and installed when you installed Net-SMNP.

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 159

The installed libraries will be found in **/lib64** or **/usr/lib64** or **/usr/local/lib64**. They may be installed under names like **libnetsnmp.so.5.1.2**. The GPFS subagent expects to find them without the appended version information in the name. Library installation should create these symbolic links for you, so you will rarely need to create them yourself. You can ensure that symbolic links exist to the versioned name from the plain name. For example,

```
# cd /usr/lib64
# ln -s libnetsnmpmibs.so.5.1.2 libnetsnmpmibs.so
```

Repeat this process for all the libraries listed in this topic.

Note: For possible Linux platform and Net-SNMP version compatibility restrictions, see the GPFS README and the GPFS FAQ in the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) or GPFS FAQ in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfsclustersfaq.html).

Configuring Net-SNMP

The GPFS subagent process connects to the Net-SNMP master agent, **snmpd**.

The following entries are required in the **snmpd** configuration file on the collector node (usually, **/etc/snmp/snmpd.conf**):

```
master agentx
AgentXSocket tcp:localhost:705
trap2sink managementhost
```

where:

managementhost
: Is the host name or IP address of the host to which you want SNMP traps sent.

If your GPFS cluster has a large number of nodes or a large number of file systems for which information must be collected, you must increase the timeout and retry parameters for communication between the SNMP master agent and the GPFS subagent to allow time for the volume of information to be transmitted. The **snmpd** configuration file entries for this are:

```
agentXTimeout 60
agentXRetries 10
```

where:

agentXTimeout
: Is set to 60 seconds for subagent to master agent communication.

agentXRetries
: Is set to 10 for the number of communication retries.

Note: Other values may be appropriate depending on the number of nodes and file systems in your GPFS cluster.

After modifying the configuration file, restart the SNMP daemon.

Configuring management applications

To configure any SNMP-based management applications you might be using (such as Tivoli NetView® or Tivoli Netcool®, or others), you must make the GPFS MIB file available on the processor on which the management application runs.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 160

You must also supply the management application with the host name or IP address of the collector node to be able to extract GPFS monitoring information through SNMP. To do this, you must be familiar with your SNMP-based management applications.

For more information about Tivoli NetView or Tivoli Netcool, see the documentation at the following information centers:

- Tivoli NetView documentation (http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?toc=/com.ibm.itnetview.doc/toc.xml)
- IBM Tivoli Network Management documentation (http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/index.jsp?toc=/com.ibm.netcool_precision.doc/toc.xml)

Installing MIB files on the collector node and management node

The GPFS management information base (MIB) file is found on the collector node in the **/usr/lpp/mmfs/data** directory with the name **GPFS-MIB.txt**.

To install this file on the collector node, do the following:

1. Copy or link the **/usr/lpp/mmfs/data/GPFS-MIB.txt** MIB file into the **SNMP MIB** directory (usually, **/usr/share/snmp/mibs**).

   Alternatively, you could add the following line to the **snmp.conf** file (usually found in the directory **/etc/snmp**):

   ```
   mibdirs +/usr/lpp/mmfs/data
   ```

2. Add the following entry to the **snmp.conf** file (usually found in the directory **/etc/snmp**):

   ```
   mibs +GPFS-MIB
   ```

3. Restart the SNMP daemon.

Different management applications have different locations and ways for installing and loading a new MIB file. The following steps for installing the GPFS MIB file apply only to Net-SNMP. If you are using other management applications, such as NetView and NetCool, refer to corresponding product manuals (listed in "Configuring management applications" on page 138) for the procedure of MIB file installation and loading.

1. Remotely copy the **/usr/lpp/mmfs/data/GPFS-MIB.txt** MIB file from the collector node into the **SNMP MIB** directory (usually, **/usr/share/snmp/mibs**).
2. Add the following entry to the **snmp.conf** file (usually found in the directory **/etc/snmp**):

   ```
   mibs +GPFS-MIB
   ```

3. You might need to restart the SNMP management application. Other steps might be necessary to make the GPFS MIB available to your management application.

Collector node administration

Collector node administration includes: assigning, unassigning, and changing collector nodes. You can also see if a collector node is defined.

To assign a collector node and start the SNMP agent, enter:

```
mmchnode --snmp-agent -N NodeName
```

To unassign a collector node and stop the SNMP agent, enter:

```
mmchnode --nosnmp-agent -N NodeName
```

To see if there is a GPFS SNMP subagent collector node defined, enter:

```
mmlscluster |grep snmp
```

To change the collector node, issue the following two commands:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 161

```
mmchnode --nosnmp-agent -N OldNodeName

mmchnode --snmp-agent -N NewNodeName
```

Starting and stopping the SNMP subagent

The SNMP subagent is started and stopped automatically.

The SNMP subagent is started automatically when GPFS is started on the collector node. If GPFS is already running when the collector node is assigned, the **mmchnode** command will automatically start the SNMP subagent.

The SNMP subagent is stopped automatically when GPFS is stopped on the node (**mmshutdown**) or when the SNMP collector node is unassigned (**mmchnode**).

The management and monitoring subagent

The GPFS SNMP management and monitoring subagent runs under an SNMP master agent such as Net-SNMP. It handles a portion of the SNMP OID space.

The management and monitoring subagent connects to the GPFS daemon on the collector node to retrieve updated information about the status of the GPFS cluster.

SNMP data can be retrieved using an SNMP application such as Tivoli NetView. NetView provides a MIB browser for retrieving user-requested data, as well as an event viewer for displaying asynchronous events.

Information that is collected includes status, configuration, and performance data about GPFS clusters, nodes, disks, file systems, storage pools, and asynchronous events. The following is a sample of the data that is collected for each of the following categories:

- Cluster status and configuration (see "Cluster status information" on page 141 and "Cluster configuration information" on page 142)
  - Name
  - Number of nodes
  - Primary and secondary servers
- Node status and configuration (see "Node status information" on page 142 and "Node configuration information" on page 142)
  - Name
  - Current status
  - Type
  - Platform
- File system status and performance (see "File system status information" on page 143 and "File system performance information" on page 144)
  - Name
  - Status
  - Total space
  - Free space
  - Accumulated statistics
- Storage pools (see "Storage pool information" on page 145)
  - Name
  - File system to which the storage pool belongs
  - Total storage pool space
  - Free storage pool space
  - Number of disks in the storage pool

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 162

- Disk status, configuration, and performance (see “Disk status information” on page 145, “Disk configuration information” on page 146, and “Disk performance information” on page 146)
  - Name
  - Status
  - Total space
  - Free space
  - Usage (metadata/data)
  - Availability
  - Statistics
- Asynchronous events (traps) (see “Net-SNMP traps” on page 147)
  - File system mounted or unmounted
  - Disks added, deleted, or changed
  - Node failure or recovery
  - File system creation, deletion, or state change
  - Storage pool is full or nearly full

Note: If file systems are not mounted on the collector node at the time that an SNMP request is received, the subagent can still obtain a list of file systems, storage pools, and disks, but some information, such as performance statistics, will be missing.

SNMP object IDs

The management and monitoring SNMP subagent serves the OID space defined as **ibm.ibmProd.ibmGPFS**, which is the numerical **enterprises.2.6.212** OID space.

Underneath this top-level space are the following:

- **gpfsTraps** at **ibmGPFS.0**
- **gpfsMIBObjects** at **ibmGPFS.1**

MIB objects

gpfsMIBObjects provides a space of objects that can be retrieved using a MIB browser application. Net-SNMP provides the **snmpget**, **snmpgetnext**, **snmptable**, and **snmpwalk** commands, which can be used to retrieve the contents of these fields.

Cluster status information

Table 22 shows the current status information for the GPFS cluster:

Table 22. gpfsClusterStatusTable: Cluster status information

Value	Description
gpfsClusterName	The cluster name.
gpfsClusterId	The cluster ID.
gpfsClusterMinReleaseLevel	The currently enabled cluster functionality level.
gpfsClusterNumNodes	The number of nodes that belong to the cluster.
gpfsClusterNumFileSystems	The number of file systems that belong to the cluster.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 163

Cluster configuration information

Table 23 shows the GPFS cluster configuration information:

Table 23. gpfsClusterConfigTable: Cluster configuration information

Value	Description
gpfsClusterConfigName	The cluster name.
gpfsClusterUidDomain	The UID domain name for the cluster.
gpfsClusterRemoteShellCommand	The remote shell command being used.
gpfsClusterRemoteFileCopyCommand	The remote file copy command being used.
gpfsClusterPrimaryServer	The primary GPFS cluster configuration server.
gpfsClusterSecondaryServer	The secondary GPFS cluster configuration server.
gpfsClusterMaxBlockSize	The maximum file system block size.
gpfsClusterDistributedTokenServer	Indicates whether the distributed token server is enabled.
gpfsClusterFailureDetectionTime	The desired time for GPFS to react to a node failure.
gpfsClusterTCPPort	The TCP port number.
gpfsClusterMinMissedPingTimeout	The lower bound on a missed **ping** timeout (seconds).
gpfsClusterMaxMissedPingTimeout	The upper bound on missed **ping** timeout (seconds).

Node status information

Table 24 shows the collected status data for each node:

Table 24. gpfsNodeStatusTable: Node status information

Node	Description
gpfsNodeName	The node name used by the GPFS daemon.
gpfsNodeIp	The node IP address.
gpfsNodePlatform	The operating system being used.
gpfsNodeStatus	The node status (for example, up or down).
gpfsNodeFailureCount	The number of node failures.
gpfsNodeThreadWait	The longest hung thread's wait time (milliseconds).
gpfsNodeHealthy	Indicates whether the node is healthy in terms of hung threads. If there are hung threads, the value is no.
gpfsNodeDiagnosis	Shows the number of hung threads and detail on the longest hung thread.
gpfsNodeVersion	The GPFS product version of the currently running daemon.

Node configuration information

Table 25 shows the collected configuration data for each node:

Table 25. gpfsNodeConfigTable: Node configuration information

Node	Description
gpfsNodeConfigName	The node name used by the GPFS daemon.
gpfsNodeType	The node type (for example, manager/client or quorum/nonquorum).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 164

Table 25. gpfsNodeConfigTable: Node configuration information (continued)

Node	Description
gpfsNodeAdmin	Indicates whether the node is one of the preferred admin nodes.
gpfsNodePagePoolL	The size of the cache (low 32 bits).
gpfsNodePagePoolH	The size of the cache (high 32 bits).
gpfsNodePrefetchThreads	The number of prefetch threads.
gpfsNodeMaxMbps	An estimate of how many megabytes of data can be transferred per second.
gpfsNodeMaxFilesToCache	The number of inodes to cache for recently-used files that have been closed.
gpfsNodeMaxStatCache	The number of inodes to keep in the stat cache.
gpfsNodeWorker1Threads	The maximum number of worker threads that can be started.
gpfsNodeDmapiEventTimeout	The maximum time the file operation threads will block while waiting for a DMAPI synchronous event (milliseconds).
gpfsNodeDmapiMountTimeout	The maximum time that the mount operation will wait for a disposition for the mount event to be set (seconds).
gpfsNodeDmapiSessFailureTimeout	The maximum time the file operation threads will wait for the recovery of the failed DMAPI session (seconds).
gpfsNodeNsdServerWaitTimeWindowOnMount	Specifies a window of time during which a mount can wait for NSD servers to come up (seconds).
gpfsNodeNsdServerWaitTimeForMount	The maximum time that the mount operation will wait for NSD servers to come up (seconds).
gpfsNodeUnmountOnDiskFail	Indicates how the GPFS daemon will respond when a disk failure is detected. If it is "true", any disk failure will cause only the local node to forcibly unmount the file system that contains the failed disk.

File system status information

Table 26 shows the collected status information for each file system:

Table 26. gpfsFileSystemStatusTable: File system status information

Value	Description
gpfsFileSystemName	The file system name.
gpfsFileSystemStatus	The status of the file system.
gpfsFileSystemXstatus	The executable status of the file system.
gpfsFileSystemTotalSpaceL	The total disk space of the file system in kilobytes (low 32 bits).
gpfsFileSystemTotalSpaceH	The total disk space of the file system in kilobytes (high 32 bits).
gpfsFileSystemNumTotalInodesL	The total number of file system inodes (low 32 bits).
gpfsFileSystemNumTotalInodesH	The total number of file system inodes (high 32 bits).
gpfsFileSystemFreeSpaceL	The free disk space of the file system in kilobytes (low 32 bits).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 165

Table 26. gpfsFileSystemStatusTable: File system status information (continued)

Value	Description
gpfsFileSystemFreeSpaceH	The free disk space of the file system in kilobytes (high 32 bits).
gpfsFileSystemNumFreeInodesL	The number of free file system inodes (low 32 bits).
gpfsFileSystemNumFreeInodesH	The number of free file system inodes (high 32 bits).

File system performance information

Table 27 shows the file system performance information:

Table 27. gpfsFileSystemPerfTable: File system performance information

Value	Description
gpfsFileSystemPerfName	The file system name.
gpfsFileSystemBytesReadL	The number of bytes read from disk, not counting those read from cache (low 32 bits).
gpfsFileSystemBytesReadH	The number of bytes read from disk, not counting those read from cache (high 32 bits).
gpfsFileSystemBytesCacheL	The number of bytes read from the cache (low 32 bits).
gpfsFileSystemBytesCacheH	The number of bytes read from the cache (high 32 bits).
gpfsFileSystemBytesWrittenL	The number of bytes written, to both disk and cache (low 32 bits).
gpfsFileSystemBytesWrittenH	The number of bytes written, to both disk and cache (high 32 bits).
gpfsFileSystemReads	The number of read operations supplied from disk.
gpfsFileSystemCaches	The number of read operations supplied from cache.
gpfsFileSystemWrites	The number of write operations to both disk and cache.
gpfsFileSystemOpenCalls	The number of file system open calls.
gpfsFileSystemCloseCalls	The number of file system close calls.
gpfsFileSystemReadCalls	The number of file system read calls.
gpfsFileSystemWriteCalls	The number of file system write calls.
gpfsFileSystemReaddirCalls	The number of file system readdir calls.
gpfsFileSystemInodesWritten	The number of inode updates to disk.
gpfsFileSystemInodesRead	The number of inode reads.
gpfsFileSystemInodesDeleted	The number of inode deletions.
gpfsFileSystemInodesCreated	The number of inode creations.
gpfsFileSystemStatCacheHit	The number of stat cache hits.
gpfsFileSystemStatCacheMiss	The number of stat cache misses.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 166

Storage pool information

Table 28 shows the collected information for each storage pool:

Table 28. gpfsStgPoolTable: Storage pool information

Value	Description
gpfsStgPoolName	The name of the storage pool.
gpfsStgPoolFSName	The name of the file system to which the storage pool belongs.
gpfsStgPoolTotalSpaceL	The total disk space in the storage pool in kilobytes (low 32 bits).
gpfsStgPoolTotalSpaceH	The total disk space in the storage pool in kilobytes (high 32 bits).
gpfsStgPoolFreeSpaceL	The free disk space in the storage pool in kilobytes (low 32 bits).
gpfsStgPoolFreeSpaceH	The free disk space in the storage pool in kilobytes (high 32 bits).
gpfsStgPoolNumDisks	The number of disks in the storage pool.

Disk status information

Table 29 shows the collected status information for each disk:

Table 29. gpfsDiskStatusTable: Disk status information

Value	Description
gpfsDiskName	The disk name.
gpfsDiskFSName	The name of the file system to which the disk belongs.
gpfsDiskStgPoolName	The name of the storage pool to which the disk belongs.
gpfsDiskStatus	The status of a disk (values: NotInUse, InUse, Suspended, BeingFormatted, BeingAdded, BeingEmptied, BeingDeleted, BeingDeleted-p, ReferencesBeingRemoved, BeingReplaced or Replacement).
gpfsDiskAvailability	The availability of the disk (Unchanged, OK, Unavailable, Recovering).
gpfsDiskTotalSpaceL	The total disk space in kilobytes (low 32 bits).
gpfsDiskTotalSpaceH	The total disk space in kilobytes (high 32 bits).
gpfsDiskFullBlockFreeSpaceL	The full block (unfragmented) free space in kilobytes (low 32 bits).
gpfsDiskFullBlockFreeSpaceH	The full block (unfragmented) free space in kilobytes (high 32 bits).
gpfsDiskSubBlockFreeSpaceL	The sub-block (fragmented) free space in kilobytes (low 32 bits).
gpfsDiskSubBlockFreeSpaceH	The sub-block (fragmented) free space in kilobytes (high 32 bits).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 167

Disk configuration information

Table 30 shows the collected disk configuration information for each disk:

Table 30. gpfsDiskConfigTable: Disk configuration information

Value	Description
gpfsDiskConfigName	The disk name.
gpfsDiskConfigFSName	The name of the file system to which the disk belongs.
gpfsDiskConfigStgPoolName	The name of the storage pool to which the disk belongs.
gpfsDiskMetadata	Indicates whether the disk holds metadata.
gpfsDiskData	Indicates whether the disk holds data.

Disk performance information

Table 31 shows the collected disk performance information for each disk:

Table 31. gpfsDiskPerfTable: Disk performance information

Value	Description
gpfsDiskPerfName	The disk name.
gpfsDiskPerfFSName	The name of the file system to which the disk belongs.
gpfsDiskPerfStgPoolName	The name of the storage pool to which the disk belongs.
gpfsDiskReadTimeL	The total time spent waiting for disk read operations (low 32 bits).
gpfsDiskReadTimeH	The total time spent waiting for disk read operations (high 32 bits).
gpfsDiskWriteTimeL	The total time spent waiting for disk write operations in microseconds (low 32 bits).
gpfsDiskWriteTimeH	The total time spent waiting for disk write operations in microseconds (high 32 bits).
gpfsDiskLongestReadTimeL	The longest disk read time in microseconds (low 32 bits).
gpfsDiskLongestReadTimeH	The longest disk read time in microseconds (high 32 bits).
gpfsDiskLongestWriteTimeL	The longest disk write time in microseconds (low 32 bits).
gpfsDiskLongestWriteTimeH	The longest disk write time in microseconds (high 32 bits).
gpfsDiskShortestReadTimeL	The shortest disk read time in microseconds (low 32 bits).
gpfsDiskShortestReadTimeH	The shortest disk read time in microseconds (high 32 bits).
gpfsDiskShortestWriteTimeL	The shortest disk write time in microseconds (low 32 bits).
gpfsDiskShortestWriteTimeH	The shortest disk write time in microseconds (high 32 bits).
gpfsDiskReadBytesL	The number of bytes read from the disk (low 32 bits).
gpfsDiskReadBytesH	The number of bytes read from the disk (high 32 bits).
gpfsDiskWriteBytesL	The number of bytes written to the disk (low 32 bits).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 168

Table 31. gpfsDiskPerfTable: Disk performance information (continued)

Value	Description
gpfsDiskWriteBytesH	The number of bytes written to the disk (high 32 bits).
gpfsDiskReadOps	The number of disk read operations.
gpfsDiskWriteOps	The number of disk write operations.

Net-SNMP traps

Traps provide asynchronous notification to the SNMP application when a particular event has been triggered in GPFS. Table 32 shows the trap types that are defined:

Table 32. Net-SNMP traps

Net-SNMP trap type	This event is triggered by:
Mount	By the mounting node when the file system is mounted on a node.
Unmount	By the unmounting node when the file system is unmounted on a node.
Add Disk	By the file system manager when a disk is added to a file system on a node.
Delete Disk	By the file system manager when a disk is deleted from a file system.
Change Disk	By the file system manager when the status of a disk or the availability of a disk is changed within the file system.
SGMGR Takeover	By the cluster manager when a file system manager takeover is successfully completed for the file system.
Node Failure	By the cluster manager when a node fails.
Node Recovery	By the cluster manager when a node recovers normally.
File System Creation	By the file system manager when a file system is successfully created.
File System Deletion	By the file system manager when a file system is deleted.
File System State Change	By the file system manager when the state of a file system changes.
New Connection	When a new connection thread is established between the events exporter and the management application.
Event Collection Buffer Overflow	By the collector node when the internal event collection buffer in the GPFS daemon overflows.
Hung Thread	By the affected node when a hung thread is detected. The GPFS Events Exporter Watchdog thread periodically checks for threads that have been waiting for longer than a threshold amount of time.
Storage Pool Utilization	By the file system manager when the utilization of a storage pool becomes full or almost full.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 169

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 170

Chapter 11. Identity management on Windows

GPFS allows file sharing among AIX, Linux, and Windows nodes. AIX and Linux rely on 32-bit user and group IDs for file ownership and access control purposes, while Windows uses variable-length security identifiers (SIDs). The difference in the user identity description models presents a challenge to any subsystem that allows for heterogeneous file sharing.

GPFS uses 32-bit ID name space as the canonical name space, and Windows SIDs are mapped into this name space as needed. Two different mapping algorithms are used (depending on system configuration):

- GPFS built-in auto-generated mapping
- User-defined mappings stored in the Microsoft Windows Active Directory using the Microsoft Identity Management for UNIX (IMU) component

Auto-generated ID mappings

Auto-generated ID mappings are the default. If no explicit mappings are created by the system administrator in the Active Directory using Microsoft Identity Management for UNIX (IMU), all mappings between security identifiers (SIDs) and UNIX IDs will be created automatically using a reserved range in UNIX ID space.

Unless the default reserved ID range overlaps with an ID already in use, no further configuration is needed to use the auto-generated mapping function. If you have a specific file system or subtree that are only accessed by user applications from Windows nodes (even if AIX or Linux nodes are used as NSD servers), auto-generated mappings will be sufficient for all application needs.

The default reserved ID range used by GPFS starts with ID 15,000,000 and covers 15,000,000 IDs. The reserved range should not overlap with any user or group ID in use on any AIX or Linux nodes. To change the starting location or the size of the reserved ID range, use the following GPFS configuration parameters:

sidAutoMapRangeLength
Controls the length of the reserved range for Windows SID to UNIX ID mapping.

sidAutoMapRangeStart
Specifies the start of the reserved range for Windows SID to UNIX ID mapping.

Note: For planning purposes, remember that auto-generated ID mappings are stored permanently with file system metadata. A change in the **sidAutoMapRangeStart** value is only effective for file systems created after the configuration change.

Installing Windows IMU

The Identity Management for UNIX (IMU) feature is included in Windows Server. This feature needs to be installed on the primary domain controller, as well as on any backup domain controllers. It is not installed by default. There are two components that need to be installed in order for IMU to function correctly.

When Active Directory is running on Windows Server 2008, follow these steps to add the IMU service:

1. Open Server Manager.
2. Under Roles, select **Active Directory Domain Services** .
3. Under Role Services, select **Add Role Services**.

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 171

4. Under the Identity Management for UNIX role service, check **Server for Network Information Services** .
5. Click **Next**, then **Install** .
6. Restart the system when the installation completes.

Configuring ID mappings in IMU

To configure ID mappings in Microsoft Identity Management for UNIX (IMU), follow the steps in this procedure.

1. Open **Active Directory Users and Computers** (accessible under Administrative Tools).
2. Select the **Users** branch in the tree on the left under the branch for your domain to see the list of users and groups in this domain.
3. Double-click on any user or group line to bring up the Properties window. If IMU is set up correctly, there will be a **UNIX Attributes** tab as shown in Figure 11:

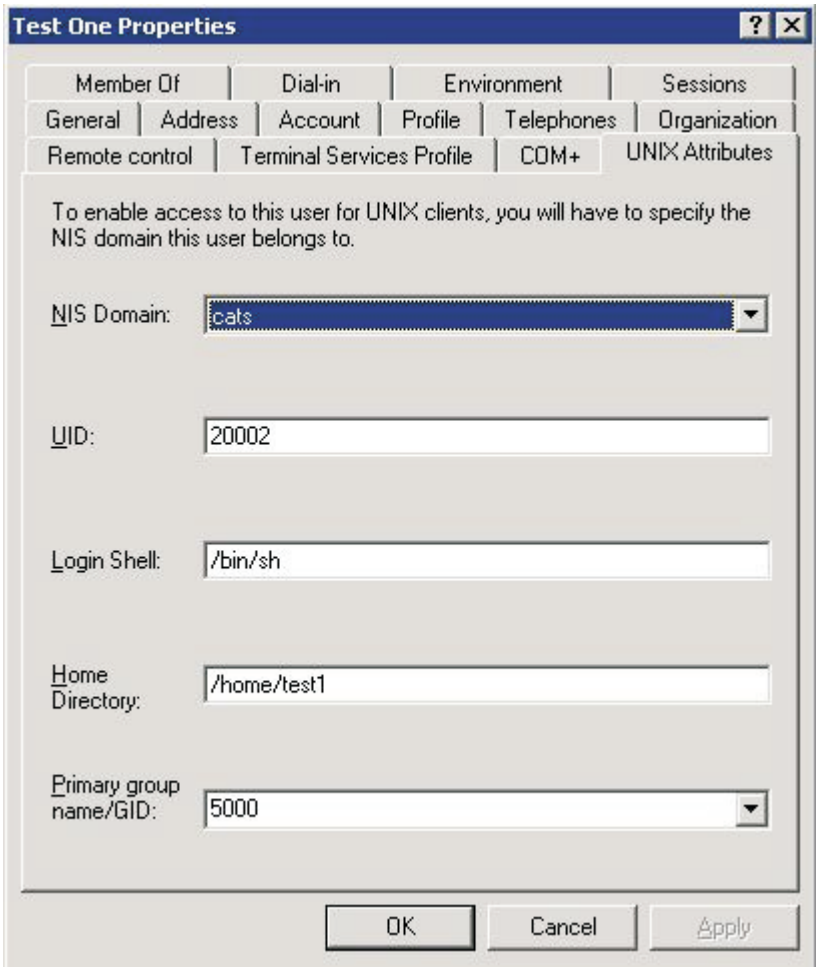


Figure 11. Properties window

Note: Because the IMU subsystem was originally designed to support integration with the UNIX Network Information Service (NIS), there is an **NIS Domain** field in the Properties window. You do not need to have NIS set up on the UNIX side. For GPFS, the NIS language does not matter.

Update information on the **UNIX Attributes** panel as follows:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 172

1. Under the **NIS Domain** drop-down list, select the name of your Active Directory domain. Selecting **<none>** will remove an existing mapping.
2. Specify a UID in the **UID** field, and for Group objects, specify a GID. This will create a bidirectional mapping between the corresponding SID and a UNIX ID. IMU will disallow the use of the same UID or GID for more than one user or group to ensure that all mappings are unique. In addition to creating mappings for domain users and groups, you can create mappings for certain built-in accounts by going to the **Builtin branch** in the Active Directory Users and Computers panel.
3. Disregard the **Primary group name/GID** field because GPFS does not use it.

It is generally better to configure all ID mappings before mounting a GPFS file system for the first time. Doing that ensures that GPFS only stores properly remapped IDs on disk. However, it is possible to add or delete mappings at any time while GPFS file systems are mounted. GPFS picks up mapping changes dynamically (the code currently checks for mapping changes every 60 seconds), and will start using them at that time.

If an IMU mapping is configured for an ID that is already recorded in some file metadata, you must proceed with caution to avoid user confusion and access disruption. Auto-generated mappings already stored in access control lists (ACLs) on disk will continue to map correctly to Windows SIDs. However, because the SID is now mapped to a different UNIX ID, when you access a file with an ACL containing its auto-generated ID, this access will effectively appear to GPFS as an access by a different user. Depending on the file access permissions, you might not be able to access files that were previously accessible. Rewriting affected ACLs after setting up a new mapping will help replace auto-generated IDs with IMU-mapped IDs, and will restore proper file access for the affected ID (this operation might need to be performed by the system administrator). Examining file ownership and permission information from a UNIX node (for example, using the **mmgetacl** command) is the easiest way to determine whether the ACL for a specified file contains auto-generated or IMU-mapped IDs.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 173

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 174

Chapter 12. GPFS Native RAID (GNR)

Note: GPFS Native RAID (GNR) is only available on the following:

- Power® 775 Disk Enclosure
- IBM System x® GPFS Storage Server (GSS). GSS is a high-capacity, high-performance storage solution that combines IBM System x servers, storage enclosures, and drives; software, including GPFS Native RAID; and networking components. GSS uses a building-block approach to create highly scalable storage for use in a broad range of application environments.

GPFS Native RAID is a software implementation of storage RAID technologies within GPFS. Using conventional dual-ported disks in a JBOD configuration, GPFS Native RAID implements sophisticated data placement and error correction algorithms to deliver high levels of storage reliability, availability, and performance. Standard GPFS file systems are created from the NSDs defined through GPFS Native RAID.

This chapter describes the basic concepts, advantages, and motivations behind GPFS Native RAID: redundancy codes; end-to-end checksums; data declustering; and administrator configuration, including recovery groups, declustered arrays, virtual disks, and virtual disk NSDs.

Overview

GPFS Native RAID integrates the functionality of an advanced storage controller into the GPFS NSD server. Unlike an external storage controller, where configuration, LUN definition, and maintenance are beyond the control of GPFS, GPFS Native RAID takes ownership of a JBOD array to directly match LUN definition, caching, and disk behavior to GPFS file system requirements.

Sophisticated data placement and error correction algorithms deliver high levels of storage reliability, availability, serviceability, and performance. GPFS Native RAID provides a variation of the GPFS network shared disk (NSD) called a virtual disk, or *vdisk*. Standard NSD clients transparently access the vdisk NSDs of a file system using the conventional NSD protocol.

The features of GPFS Native RAID include:

- Software RAID: GPFS Native RAID runs on standard AIX disks in a dual-ported JBOD array, which does not require external RAID storage controllers or other custom hardware RAID acceleration.
- Declustering: GPFS Native RAID distributes client data, redundancy information, and spare space uniformly across all disks of a JBOD. This distribution reduces the rebuild (disk failure recovery process) overhead compared to conventional RAID.
- Checksum: An end-to-end data integrity check, using checksums and version numbers, is maintained between the disk surface and NSD clients. The checksum algorithm uses version numbers to detect silent data corruption and lost disk writes.
- Data redundancy: GPFS Native RAID supports highly reliable 2-fault-tolerant and 3-fault-tolerant Reed-Solomon based parity codes and 3-way and 4-way replication.
- Large cache: A large cache improves read and write performance, particularly for small I/O operations.
- Arbitrarily sized disk arrays: The number of disks is not restricted to a multiple of the RAID redundancy code width, which allows flexibility in the number of disks in the RAID array.
- Multiple redundancy schemes: One disk array can support vdisks with different redundancy schemes, for example Reed-Solomon and replication codes.
- Disk hospital: A disk hospital asynchronously diagnoses faulty disks and paths, and requests replacement of disks by using past health records.
- Automatic recovery: Seamlessly and automatically recovers from primary server failure.

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 175

- Disk scrubbing: A disk scrubber automatically detects and repairs latent sector errors in the background.
- Familiar interface: Standard GPFS command syntax is used for all configuration commands; including, maintaining and replacing failed disks.
- Flexible hardware configuration: Support of JBOD enclosures with multiple disks physically mounted together on removable carriers.
- Configuration and data logging: Internal configuration and small-write data are automatically logged to solid-state disks for improved performance.

GPFS Native RAID features

This section introduces three key features of GPFS Native RAID and how they work: data redundancy using RAID codes, end-to-end checksums, and declustering.

RAID codes

GPFS Native RAID corrects for disk failures and other storage faults automatically by reconstructing the unreadable data using the available data redundancy of a Reed-Solomon code or *N*-way replication. GPFS Native RAID uses the reconstructed data to fulfill client operations, and in the case of disk failure, to rebuild the data onto spare space. GPFS Native RAID supports 2- and 3-fault-tolerant Reed-Solomon codes and 3-way and 4-way replication, which respectively detect and correct up to two or three concurrent faults[1]. The redundancy code layouts that GPFS Native RAID supports, called *tracks*, are illustrated in Figure 12.

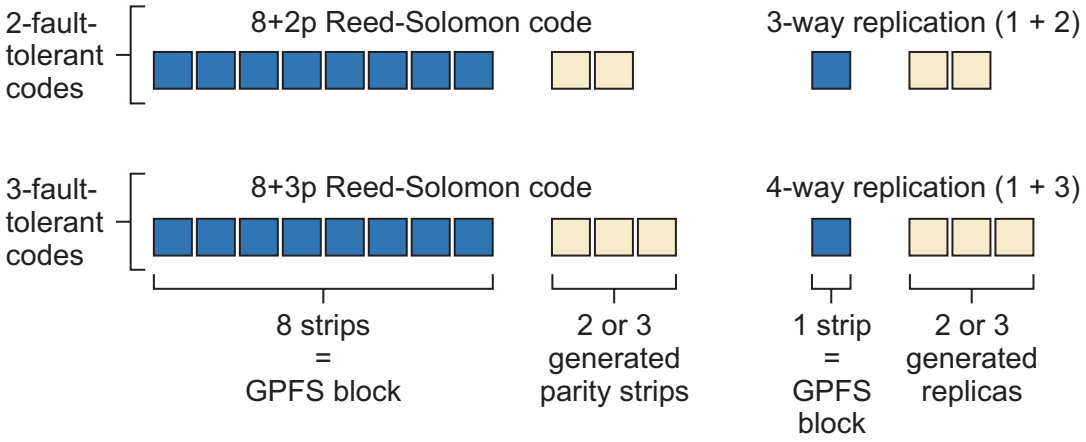


Figure 12. Redundancy codes supported by GPFS Native RAID. GPFS Native RAID supports 2- and 3-fault-tolerant Reed-Solomon codes, which partition a GPFS block into eight data strips and two or three parity strips. The *N*-way replication codes duplicate the GPFS block on *N* - 1 replica strips.

Depending on the configured RAID code, GPFS Native RAID creates redundancy information automatically. Using a Reed-Solomon code, GPFS Native RAID divides a GPFS block of user data equally into eight *data strips* and generates two or three redundant *parity strips.* This results in a stripe or track width of 10 or 11 strips and storage efficiency of 80% or 73%, respectively (excluding user-configurable spare space for rebuild operations).

Using *N*-way replication, a GPFS data block is replicated simply $N - 1$ times, in effect implementing 1 + 2 and 1 + 3 redundancy codes, with the strip size equal to the GPFS block size. Thus, for every block/strip

1. An f-fault-tolerant Reed-Solomon code or a $(1 + f)$-way replication can survive the *concurrent* failure of f disks or read faults. Also, if there are s equivalent spare disks in the array, an f-fault-tolerant array can survive the *sequential* failure of $f + s$ disks where disk failures occur between successful rebuild operations.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 176

that is written to the disks, *N* replicas of that block/strip are also written. This results in a track width of three or four strips and storage efficiency of 33% or 25%, respectively.

End-to-end checksum

Most implementations of RAID codes implicitly assume that disks reliably detect and report faults, hard-read errors, and other integrity problems. However, studies have shown that disks do not report some read faults and occasionally fail to write data, while actually claiming to have written the data. These errors are often referred to as silent errors, phantom-writes, dropped-writes, and off-track writes. To cover for these shortcomings, GPFS Native RAID implements an end-to-end checksum that can detect silent data corruption caused by either disks or other system components that transport or manipulate the data.

When an NSD client is writing data, a checksum of 8 bytes is calculated and appended to the data before it is transported over the network to the GPFS Native RAID server. On reception, GPFS Native RAID calculates and verifies the checksum. Then, GPFS Native RAID stores the data, a checksum, and version number to disk and logs the version number in its metadata for future verification during read.

When GPFS Native RAID reads disks to satisfy a client read operation, it compares the disk checksum against the disk data and the disk checksum version number against what is stored in its metadata. If the checksums and version numbers match, GPFS Native RAID sends the data along with a checksum to the NSD client. If the checksum or version numbers are invalid, GPFS Native RAID reconstructs the data using parity or replication and returns the reconstructed data and a newly generated checksum to the client. Thus, both silent disk read errors and lost or missing disk writes are detected and corrected.

Declustered RAID

Compared to conventional RAID, GPFS Native RAID implements a sophisticated data and spare space disk layout scheme that allows for arbitrarily sized disk arrays while also reducing the overhead to clients when recovering from disk failures. To accomplish this, GPFS Native RAID uniformly spreads or *declusters* user data, redundancy information, and spare space across all the disks of a declustered array. Figure 13 on page 156 compares a conventional RAID layout versus an equivalent declustered array.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 177

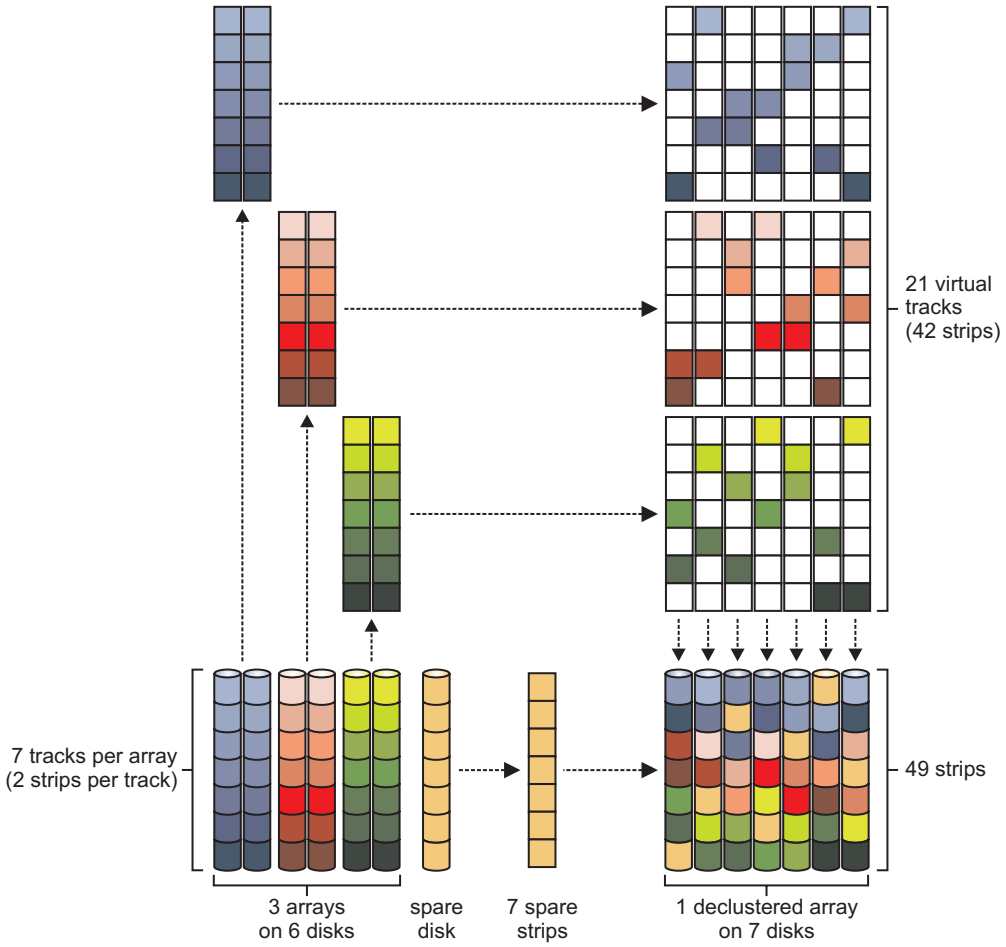


Figure 13. Conventional RAID versus declustered RAID layouts. This figure is an example of how GPFS Native RAID improves client performance during rebuild operations by utilizing the throughput of all disks in the declustered array. This is illustrated here by comparing a conventional RAID of three arrays versus a declustered array, both using 7 disks. A conventional 1-fault-tolerant 1 + 1 replicated RAID array in the lower left is shown with three arrays of two disks each (data and replica strips) and a spare disk for rebuilding. To decluster this array, the disks are divided into seven tracks, two strips per array, as shown in the upper left. The strips from each group are then combinatorially spread across all seven disk positions, for a total of 21 virtual tracks, per the upper right. The strips of each disk position for every track are then arbitrarily allocated onto the disks of the declustered array of the lower right (in this case, by vertically sliding down and compacting the strips from above). The spare strips are uniformly inserted, one per disk.

As illustrated in Figure 14 on page 157, a declustered array can significantly shorten the time required to recover from a disk failure, which lowers the rebuild overhead for client applications. When a disk fails, erased data is rebuilt using all the operational disks in the declustered array, the bandwidth of which is greater than that of the fewer disks of a conventional RAID group. Furthermore, if an additional disk fault occurs during a rebuild, the number of impacted tracks requiring repair is markedly less than the previous failure and less than the constant rebuild overhead of a conventional array.

The decrease in declustered rebuild impact and client overhead can be a factor of three to four times less than a conventional RAID. Because GPFS stripes client data across all the storage nodes of a cluster, file system performance becomes less dependent upon the speed of any single rebuilding storage array.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 178

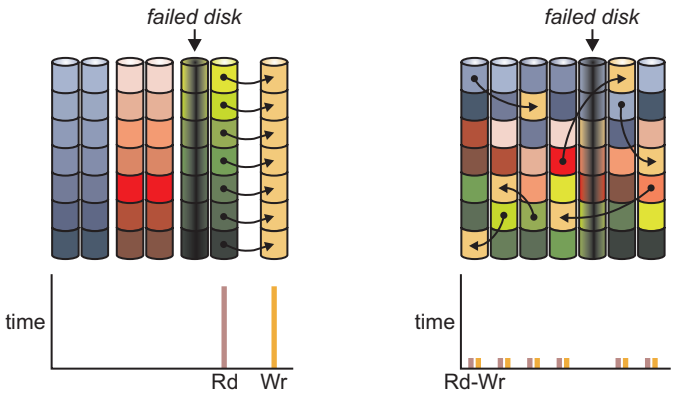


Figure 14. Lower rebuild overhead in conventional RAID versus declustered RAID. When a single disk fails in the 1-fault-tolerant 1 + 1 conventional array on the left, the redundant disk is read and copied onto the spare disk, which requires a throughput of 7 strip I/O operations. When a disk fails in the declustered array, all replica strips of the six impacted tracks are read from the surviving six disks and then written to six spare strips, for a throughput of 2 strip I/O operations. The bar chart illustrates disk read and write I/O throughput during the rebuild operations.

Disk configurations

This section describes recovery group and declustered array configurations.

Recovery groups

GPFS Native RAID divides disks into *recovery groups* where each is physically connected to two servers: primary and backup. All accesses to any of the disks of a recovery group are made through the active server of the recovery group, either the primary or backup.

Building on the inherent NSD failover capabilities of GPFS, when a GPFS Native RAID server stops operating because of a hardware fault, software fault, or normal shutdown, the backup GPFS Native RAID server seamlessly takes over control of the associated disks of its recovery groups.

Typically, a JBOD array is divided into two recovery groups controlled by different primary GPFS Native RAID servers. If the primary server of a recovery group fails, control automatically switches over to its backup server. Within a typical JBOD, the primary server for a recovery group is the backup server for the other recovery group.

Figure 15 on page 158 illustrates the ring configuration where GPFS Native RAID servers and storage JBODs alternate around a loop. A particular GPFS Native RAID server is connected to two adjacent storage JBODs and vice versa. The ratio of GPFS Native RAID server to storage JBODs is thus one-to-one. Load on servers increases by 50% when a server fails.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 179

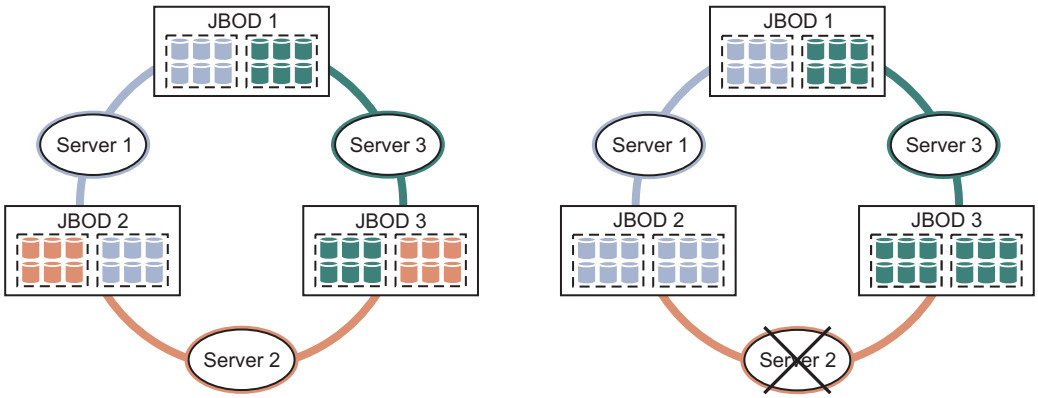


Figure 15. GPFS Native RAID server and recovery groups in a ring configuration. A recovery group is illustrated as the dashed-line enclosed group of disks within a storage JBOD. Server *N* is the primary controller of the first recovery group in JBOD *N* (and backup for its second recovery group), and the primary controller of the second recovery group in JBOD *N* + 1 (and backup for its first recovery group). As shown, when server 2 fails, control of the first recovery group in JBOD 2 is taken over by its backup server 1, and control of the second recovery group in JBOD 3 is taken over by its backup server 3. During the failure of server 2, the load on backup server 1 and 3 increases by 50% from two to three recovery groups.

For small configurations, Figure 16 illustrates a setup with two GPFS Native RAID servers connected to one storage JBOD. For handling server failures, this configuration can be less efficient for large clusters because it requires $2 \times N$ servers each capable of serving two recovery groups, where N is the number of JBOD arrays. Conversely, the ring configuration requires $1 \times N$ servers each capable of serving three recovery groups.

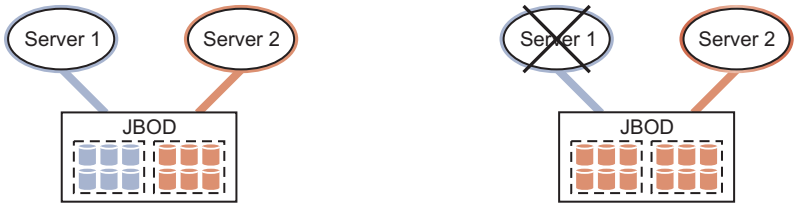


Figure 16. Minimal configuration of two GPFS Native RAID servers and one storage JBOD. GPFS Native RAID server 1 is the primary controller for the first recovery group and backup for the second recovery group. GPFS Native RAID server 2 is the primary controller for the second recovery group and backup for the first recovery group. As shown, when server 1 fails, control of the first recovery group is taken over by its backup server 2. During the failure of server 1, the load on backup server 2 increases by 100% from one to two recovery groups.

Declustered arrays

A declustered array is a subset of the physical disks (pdisks) in a recovery group across which data, redundancy information, and spare space are declustered. The number of disks in a declustered array is determined by the RAID code-width of the vdisks that will be housed in the declustered array. For more information, see "Virtual disks" on page 159. There can be one or more declustered arrays per recovery group. Figure 17 on page 159 illustrates a storage JBOD with two recovery groups, each with four declustered arrays.

A declustered array can hold one or more vdisks. Since redundancy codes are associated with vdisks, a declustered array can simultaneously contain both Reed-Solomon and replicated vdisks.

If the storage JBOD supports multiple disks physically mounted together on removable carriers, removal of a carrier temporarily disables access to all the disks in the carrier. Thus, pdisks on the same carrier should not be in the same declustered array, as vdisk redundancy protection would be weakened upon carrier removal.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 180

Declustered arrays are normally created at recovery group creation time but new ones can be created or existing ones grown by adding pdisks at a later time.

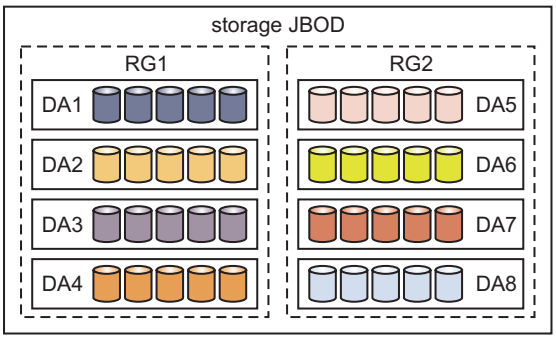


Figure 17. Example of declustered arrays and recovery groups in storage JBOD. This figure shows a storage JBOD with two recovery groups, each recovery group with four declustered arrays, and each declustered array with five disks.

Virtual and physical disks

A virtual disk (vdisk) is a type of NSD, implemented by GPFS Native RAID across all the physical disks (pdisks) of a declustered array. Multiple vdisks can be defined within a declustered array, typically Reed-Solomon vdisks for GPFS user data and replicated vdisks for GPFS metadata.

Virtual disks

Whether a vdisk of a particular capacity can be created in a declustered array depends on its redundancy code, the number of pdisks and equivalent spare capacity in the array, and other small GPFS Native RAID overhead factors. The **mmcrvdisk** command can automatically configure a vdisk of the largest possible size given a redundancy code and configured spare space of the declustered array.

In general, the number of pdisks in a declustered array cannot be less than the widest redundancy code of a vdisk plus the equivalent spare disk capacity of a declustered array. For example, a vdisk using the 11-strip-wide 8 + 3p Reed-Solomon code requires at least 13 pdisks in a declustered array with the equivalent spare space capacity of two disks. A vdisk using the 3-way replication code requires at least five pdisks in a declustered array with the equivalent spare capacity of two disks.

Vdisks are partitioned into virtual tracks, which are the functional equivalent of a GPFS block. All vdisk attributes are fixed at creation and cannot be subsequently altered.

Physical disks

A pdisk is used by GPFS Native RAID to store both user data and GPFS Native RAID internal configuration data.

A pdisk is either a conventional rotating magnetic-media disk (HDD) or a solid-state disk (SSD). All pdisks in a declustered array must have the same capacity.

It is assumed that pdisks are dual ported. This configuration means one or more paths are connected to the primary GPFS Native RAID server and one or more paths are connected to the backup server. There are typically two redundant paths between a GPFS Native RAID server and connected JBOD pdisks.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 181

Solid-state disks

GPFS Native RAID assumes several solid-state disks (SSDs) in each recovery group in order to redundantly log changes to its internal configuration and fast-write data in non-volatile memory, which is accessible from either the primary or backup GPFS Native RAID servers after server failure. A typical GPFS Native RAID log vdisk might be configured as 3-way replication over a dedicated declustered array of 4 SSDs per recovery group.

Disk hospital

The disk hospital is a key feature of GPFS Native RAID that asynchronously diagnoses errors and faults in the storage subsystem. GPFS Native RAID times out an individual pdisk I/O operation after about ten seconds, thereby limiting the impact from a faulty pdisk on a client I/O operation. When a pdisk I/O operation results in a timeout, an I/O error, or a checksum mismatch, the suspect pdisk is immediately admitted into the disk hospital. When a pdisk is first admitted, the hospital determines whether the error was caused by the pdisk itself or by the paths to it. While the hospital diagnoses the error, GPFS Native RAID, if possible, uses vdisk redundancy codes to reconstruct lost or erased strips for I/O operations that would otherwise have used the suspect pdisk.

Health metrics

The disk hospital maintains the following internal health assessment metrics for each pdisk. When one of these metrics exceeds the threshold, the pdisk is marked for replacement according to the disk maintenance replacement policy for the declustered array.

relativePerformance
: Characterizes response times. Values greater than one indicate that the disk is performing above average speed; values less than one indicate that the disk is performing below average speed. Values within a range of 0.800 to 1.250 are considered normal. Examples of typical values are: 0.932, 0.978, 1.039, and 1.095 If the **relativePerformance** of a disk falls below a particular threshold (the default setting is 0.667), the hospital adds “slow” to the pdisk state, and the disk is prepared for replacement.

dataBadness
: Characterizes media errors (hard errors) and checksum errors.

The disk hospital logs selected Self-Monitoring, Analysis and Reporting Technology (SMART) data, including the number of internal sector remapping events for each pdisk.

Pdisk discovery

GPFS Native RAID discovers all connected pdisks when it starts up, and then regularly schedules a process that will rediscover a pdisk that becomes newly accessible to the GPFS Native RAID server. This allows pdisks to be physically connected or connection problems to be repaired without restarting the GPFS Native RAID server.

Disk replacement

The disk hospital keeps track of disks that require replacement according to the disk replacement policy of the declustered array, and it can be configured to report the need for replacement in a variety of ways. It records and reports the FRU number and physical hardware location of failed disks to help guide service personnel to the correct location with replacement disks.

When multiple disks are mounted on a removable carrier, each a member of a different declustered array, disk replacement requires the hospital to temporarily suspend other disks in the same carrier. In order to

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 182

guard against human error, carriers are also not removable until GPFS Native RAID actuates a solenoid controlled latch. In response to administrative commands, the hospital quiesces the appropriate disks, releases the carrier latch, and turns on identify lights on the carrier adjacent to the disks that require replacement.

After one or more disks are replaced and the carrier is re-inserted, the hospital, in response to administrative commands, verifies that the repair has taken place and automatically adds any new disks to the declustered array, which causes GPFS Native RAID to rebalance the tracks and spare space across all the disks of the declustered array. If service personnel fail to re-insert the carrier within a reasonable period, the hospital declares the disks on the carrier as `missing` and starts rebuilding the affected data.

Managing GPFS Native RAID

This section describes, in more detail, the characteristics and behavior of GPFS Native RAID entities: recovery groups, pdisks, declustered arrays, and vdisks. Disk maintenance and overall GPFS Native RAID management are also described.

Recovery groups

A *recovery group* is the fundamental organizing structure employed by GPFS Native RAID. A recovery group is conceptually the internal GPFS equivalent of a hardware disk controller. Within a recovery group, individual JBOD disks are defined as *pdisks* and assigned to *declustered arrays*. Each pdisk belongs to exactly one declustered array within one recovery group. Within a declustered array of pdisks, *vdisks* are defined. The vdisks are the equivalent of the RAID LUNs for a hardware disk controller. One or two GPFS cluster nodes must be defined as the servers for a recovery group, and these servers must have direct hardware connections to the JBOD disks in the recovery group. Two servers are recommended for high availability server failover, but only one server will actively manage the recovery group at any given time. One server is the preferred and *primary* server, and the other server, if defined, is the *backup* server.

Multiple recovery groups can be defined, and a GPFS cluster node can be the primary or backup server for more than one recovery group. The name of a recovery group must be unique within a GPFS cluster.

Recovery group server parameters

To enable a GPFS cluster node as a recovery group server, it must have the **mmchconfig** configuration parameter **nsdRAIDTracks** set to a nonzero value, and the GPFS daemon must be restarted on the node. The **nsdRAIDTracks** parameter defines the maximum number of vdisk track descriptors that the server can have in memory at a given time. The volume of actual vdisk data that the server can cache in memory is governed by the size of the GPFS pagepool on the server and the value of the **nsdRAIDBufferPoolSizePct** configuration parameter. The **nsdRAIDBufferPoolSizePct** parameter defaults to 50% of the pagepool on the server. A recovery group server should be configured with a substantial amount of pagepool, on the order of tens of gigabytes. A recovery group server becomes an NSD server after NSDs are defined on the vdisks in the recovery group, so the **nsdBufSpace** parameter also applies. The default for **nsdBufSpace** is 30% of the pagepool, and it can be decreased to its minimum value of 10% because the vdisk data buffer pool is used directly to serve the vdisk NSDs.

The vdisk track descriptors, as governed by **nsdRAIDTracks**, include such information as the RAID code, track number, and status. The descriptors also contain pointers to vdisk data buffers in the GPFS pagepool, as governed by **nsdRAIDBufferPoolSizePct**. It is these buffers that hold the actual vdisk data and redundancy information.

For more information on how to set the **nsdRAIDTracks** and **nsdRAIDBufferPoolSizePct** parameters, see "Planning considerations for GPFS Native RAID" on page 178.

For more information on the **nsdRAIDTracks**, **nsdRAIDBufferPoolSizePct**, and **nsdBufSpace** parameters, see the **mmchconfig** command in the *GPFS: Administration and Programming Reference*.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 183

Recovery group creation

Recovery groups are created using the **mmcrrecoverygroup** command, which takes the following arguments:

- The name of the recovery group to create.
- The name of a stanza file describing the declustered arrays and pdisks within the recovery group.
- The names of the GPFS cluster nodes that will be the primary and, if specified, backup servers for the recovery group.

When a recovery group is created, the GPFS daemon must be running with the **nsdRAIDTracks** configuration parameter in effect on the specified servers.

For more information see the **mmcrrecoverygroup** command in the *GPFS: Administration and Programming Reference*.

Recovery group server failover

When, as is recommended, a recovery group is assigned two servers, one server is the preferred and *primary* server for the recovery group and the other server is the *backup* server. Only one server can serve the recovery group at any given time; this server is known as the *active* recovery group server. The server that is not currently serving the recovery group is the *standby* server. If the active recovery group server is unable to serve a recovery group, it will relinquish control of the recovery group and pass it to the standby server, if available. The failover from the active to the standby server should be transparent to any GPFS file system using the vdisk NSDs in the recovery group. There will be a pause in access to the file system data in the vdisk NSDs of the recovery group while the recovery operation takes place on the new server. This server failover recovery operation involves the new server opening the component disks of the recovery group and playing back any logged RAID transactions.

The active server for a recovery group can be changed by the GPFS administrator using the **mmchrecoverygroup** command. This command can also be used to change the primary and backup servers for a recovery group. For more information, see the **mmchrecoverygroup** command in the *GPFS: Administration and Programming Reference*.

Pdisks

The GPFS Native RAID *pdisk* is an abstraction of a physical disk. A pdisk corresponds to exactly one physical disk, and belongs to exactly one declustered array within exactly one recovery group. Before discussing how declustered arrays collect pdisks into groups, it will be useful to describe the characteristics of pdisks.

A recovery group may contain a maximum of 512 pdisks. A declustered array within a recovery group may contain a maximum of 128 pdisks. The name of a pdisk must be unique within a recovery group; that is, two recovery groups may each contain a pdisk named `disk10`, but a recovery group may not contain two pdisks named `disk10`, even if they are in different declustered arrays.

A pdisk is usually created using the **mmcrrecoverygroup** command, whereby it is assigned to a declustered array within a newly created recovery group. In unusual situations, pdisks may also be created and assigned to a declustered array of an existing recovery group by using the **mmaddpdisk** command.

To create a pdisk, a stanza must be supplied to the **mmcrrecoverygroup** or **mmaddpdisk** commands specifying the pdisk name, the declustered array name to which it is assigned, and a block device special file name for the entire physical disk as it is configured by the operating system on the active recovery group server. The following is an example pdisk creation stanza:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 184

```
%pdisk: pdiskName=c073d1
        da=DA1
        device=/dev/hdisk192
```

The device name for a pdisk must refer to the entirety of a single physical disk; pdisks should not be created using virtualized or software-based disks (for example, logical volumes, disk partitions, logical units from other RAID controllers, or network-attached disks). For a pdisk to be successfully created, the physical disk must be present and functional at the specified device name on the active server. The physical disk must also be present on the standby recovery group server, if one is configured (note that the physical disk block device special name on the standby server will almost certainly be different, and will automatically be discovered by GPFS).

The attributes of a pdisk include the physical disk's unique world wide name (WWN), its field replaceable unit (FRU) code, and its physical location code. Pdisk attributes may be displayed by the **mmlspdisk** command; of particular interest here are the pdisk device *paths* and the pdisk *states*.

Pdisks that have failed and been marked for replacement by the disk hospital are replaced using the **mmchcarrier** command. In unusual situations, pdisks may be added or deleted using the **mmaddpdisk** or **mmdelpdisk** commands. When deleted, either through replacement or the **mmdelpdisk** command, the pdisk abstraction will only cease to exist when all the data it contained has been rebuilt onto spare space (even though the physical disk may have been removed from the system).

Pdisks are normally under the control of GPFS Native RAID and the disk hospital. In some situations, however, the **mmchpdisk** command may be used to directly manipulate pdisks. For example, if a pdisk has to be temporarily removed to allow for hardware maintenance on other parts of the system, you can use the **mmchpdisk --begin-service-drain** command to drain the data before removing the pdisk. After bringing the pdisk back online, you can use the **mmchpdisk --end-service-drain** command to return the drained data to the pdisk.

Note: This process requires that there be sufficient spare space in the declustered array for the data that is to be drained. If the available spare space is insufficient, it can be increased with the **mmchrecoverygroup** command.

Pdisk paths

To the operating system, physical disks are made visible as block devices with device special file names, such as `/dev/hdisk32`. To achieve high availability and throughput, the physical disks of a JBOD array are connected to each server by multiple (usually two) interfaces in a configuration known as *multipath* (or *dualpath*). When two operating system block devices are visible for each physical disk, GPFS Native RAID refers to them as the *paths* to the pdisk.

In normal operation, the paths to individual pdisks are automatically discovered by GPFS Native RAID. There are only two instances when a pdisk must be referred to by its explicit block device path name: during recovery group creation using the **mmcrrecoverygroup** command, and when adding new pdisks to an existing recovery group with the **mmaddpdisk** command. In both of these cases, only one of the block device path names as seen on the active server needs to be specified; any other paths on the active and standby servers will be automatically discovered.

The operating system may have the ability to internally merge multiple paths to a physical disk into a single block device. When GPFS Native RAID is in use, the operating system multipath merge function must be disabled because GPFS Native RAID itself manages the individual paths to the disk. For more information, see “Example scenario: Configuring GPFS Native RAID recovery groups” on page 182.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 185

Pdisk states

GPFS Native RAID maintains its view of a pdisk and its corresponding physical disk by means of a *pdisk state*. The pdisk state consists of multiple keyword flags, which may be displayed using the **mmlsrecoverygroup** or **mmlspdisk** commands. The pdisk state flags indicate how GPFS Native RAID is currently using or managing a disk.

In normal circumstances, the state of the vast majority of pdisks will be represented by the sole keyword `ok`. This means that GPFS Native RAID considers the pdisk to be healthy: the recovery group server is able to communicate with the disk, the disk is functioning normally, and the disk can be used to store data. The `diagnosing` flag will be present in the pdisk state when the GPFS Native RAID disk hospital suspects, or attempts to correct, a problem. If GPFS Native RAID is unable to communicate with a disk, the pdisk state will include the keyword `missing`. If a `missing` disk becomes reconnected and functions properly, its state will change back to `ok`. The `readonly` flag means that a disk has indicated that it can no longer safely write data. A disk can also be marked by the disk hospital as `failing`, perhaps due to an excessive number of media or checksum errors. When the disk hospital concludes that a disk is no longer operating effectively, it will declare the disk to be `dead`. If the number of `dead` pdisks reaches or exceeds the replacement threshold of their declustered array, the disk hospital will add the flag `replace` to the pdisk state, which indicates that physical disk replacement should be performed as soon as possible.

When the state of a pdisk indicates that it can no longer behave reliably, GPFS Native RAID will rebuild the pdisk's data onto spare space on the other pdisks in the same declustered array. This is called *draining* the pdisk. That a pdisk is draining or has been drained will be indicated by a keyword in the pdisk state flags. The flag `systemDrain` means that GPFS Native RAID has decided to rebuild the data from the pdisk; the flag `adminDrain` means that the GPFS administrator issued the **mmdelpdisk** command to delete the pdisk.

GPFS Native RAID stores both user (GPFS file system) data and its own internal recovery group data and vdisk configuration data on pdisks. Additional pdisk state flags indicate when these data elements are not present on a pdisk. When a pdisk starts draining, GPFS Native RAID first replicates the recovery group data and vdisk configuration data onto other pdisks. When this completes, the flags `noRGD` (no recovery group data) and `noVCD` (no vdisk configuration data) are added to the pdisk state flags. When the slower process of removing all user data completes, the `noData` flag will be added to the pdisk state.

To summarize, the vast majority of pdisks will be in the `ok` state during normal operation. The `ok` state indicates that the disk is reachable, functioning, not draining, and that the disk contains user data and GPFS Native RAID recovery group and vdisk configuration information. A more complex example of a pdisk state is `dead/systemDrain/noRGD/noVCD/noData` for a single pdisk that has failed. This set of pdisk state flags indicates that the pdisk was declared dead by the system, was marked to be drained, and that all of its data (recovery group, vdisk configuration, and user) has been successfully rebuilt onto the spare space on other pdisks.

In addition to those discussed here, there are some transient pdisk states that have little impact on normal operations; the complete set of states is documented in Table 33.

Table 33. Pdisk states

State	Description
`ok`	The disk is functioning normally.
`dead`	The disk failed.
`missing`	GPFS Native RAID is unable to communicate with the disk.
`diagnosing`	The disk is temporarily unusable while its status is determined by the disk hospital.
`suspended`	The disk is temporarily unusable as part of a service procedure.
`readonly`	The disk is no longer writeable.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 186

Table 33. Pdisk states (continued)

State	Description
failing	The disk is not healthy but not dead.
slow	The disk's performance is far below expectations.
systemDrain	The disk is faulty, so data and configuration data must be drained.
adminDrain	An administrator requested that this pdisk be deleted.
serviceDrain	The disk is being drained to be taken temporarily out of service.
noRGD	The recovery group data was drained from the disk.
noVCD	All vdisk configuration data was drained from the disk.
noData	All vdisk user data was drained from the disk.
undrainable	There is not enough spare space in the declustered array to drain this disk.
replace	Replacement of the disk was requested.
noPath	There was no functioning path found to this disk.
PTOW	The disk is temporarily unusable because of a pending timed-out write.
init	The pdisk object is being initialized or removed.
formatting	Initial configuration data is being written to the disk.

Declustered arrays

Declustered arrays are disjoint subsets of the pdisks in a recovery group. Vdisks are created within declustered arrays, and vdisk tracks are declustered across all of an array's pdisks. A recovery group may contain up to 16 declustered arrays. A declustered array may contain up to 128 pdisks (but the total number of pdisks in all declustered arrays within a recovery group may not exceed 512). A pdisk may belong to only one declustered array. The name of a declustered array must be unique within a recovery group; that is, two recovery groups may each contain a declustered array named DA3, but a recovery group may not contain two declustered arrays named DA3. The pdisks within a declustered array must all be of the same size and should all have similar performance characteristics.

A declustered array is usually created together with its member pdisks and its containing recovery group through the use of the **mmchrecoverygroup** command. A declustered array may also be created using the **mmaddpdisk** command to add pdisks to a declustered array that does not yet exist in a recovery group. A declustered array may be deleted by deleting its last member pdisk, or by deleting the recovery group in which it resides. Any vdisk NSDs and vdisks within the declustered array must already have been deleted. There are no explicit commands to create or delete declustered arrays.

Declustered arrays serve two purposes:

- Segregating a small number of fast SSDs into their own group for storing the vdisk log (the RAID update and recovery group event log).
- Partitioning the disks of a JBOD enclosure into smaller subsets exclusive of a common point of failure, such as removable carriers that hold multiple disks.

The latter consideration comes into play when one considers that removing a disk carrier to perform disk replacement also temporarily removes some good disks, perhaps a number in excess of the fault tolerance of the vdisk NSDs. This would cause temporary suspension of file system activity until the disks are restored. To avoid this, each disk position in a removable carrier should be used to define a separate declustered array, such that disk position one defines DA1, disk position two defines DA2, and so on. Then when a disk carrier is removed, each declustered array will suffer the loss of just one disk, which is within the fault tolerance of any GPFS Native RAID vdisk NSD.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 187

Declustered array parameters

Declustered arrays have three parameters that may be changed using the **mmchrecoverygroup** command with the **--declustered-array** option. These are:

- The number of disks' worth of equivalent spare space. This defaults to one for arrays with nine or fewer pdisks, and two for arrays with 10 or more pdisks.
- The number of disks that must fail before the declustered array is marked as needing to have disks replaced. The default is the number of spares.
- The number of days over which all the vdisks in the declustered array are scrubbed for errors. The default is 14 days.

Declustered array size

GPFS Native RAID distinguishes between large and small declustered arrays. A declustered array is considered *large* if at the time of its creation it contains at least 11 pdisks, including an equivalent spare space of two disks (or at least 10 pdisks, including an equivalent spare space of one disk). All other declustered arrays are considered *small*. At least one declustered array in each recovery group must be large, because only large declustered arrays have enough pdisks to safely store an adequate number of replicas of the GPFS Native RAID configuration data for the recovery group.

Because the narrowest RAID code that GPFS Native RAID supports is 3-way replication, the smallest possible declustered array contains four pdisks, including the minimum required equivalent spare space of one disk. The RAID code width of the intended vdisk NSDs and the amount of equivalent spare space also affect declustered array size; if Reed-Solomon 8 + 3p vdisks, which have a code width of 11, are required, and two disks of equivalent spare space is also required, the declustered array must have at least 13 member pdisks.

Spare space

While operating with a failed pdisk in a declustered array, GPFS Native RAID continues to serve file system I/O requests by using redundancy information on other pdisks to reconstruct data that cannot be read, and by marking data that cannot be written to the failed pdisk as stale. Meanwhile, to restore full redundancy and fault tolerance, the data on the failed pdisk is rebuilt onto *spare space*, reserved unused portions of the declustered array that are declustered over all the member pdisks. The failed disk is thereby *drained* of its data by copying it to the spare space.

The amount of spare space in a declustered array is set at creation time and may be changed later. The spare space is expressed in whole units equivalent to the capacity of a member pdisk of the declustered array, but is spread among all the member pdisks. There are no dedicated spare pdisks. This implies that a number of pdisks equal to the specified spare space may fail, and the full redundancy of all the data in the declustered array can be restored through rebuild.

At minimum, each declustered array requires spare space equivalent to the size of one member pdisk. Because large declustered arrays have a greater probability of disk failure, the default amount of spare space depends on the size of the declustered array. A declustered array with nine or fewer pdisks defaults to having one disk of equivalent spare space. A declustered array with 10 or more disks defaults to having two disks of equivalent spare space. These defaults can be overridden, especially at declustered array creation. However, if at a later point too much of the declustered array is already allocated for use by vdisks, it may not be possible to increase the amount of spare space.

Declustered array free space

The declustered array free space reported by the **mmlsrecoverygroup** command reflects the space available for creating vdisks. Spare space is not included in this value since it is not available for creating new vdisks.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 188

Pdisk free space

The pdisk free space reported by the **mmlsrecoverygroup** command reflects the actual number of unused data partitions on the disk. This includes spare space, so if a pdisk fails, these values will decrease as data is moved to the spare space.

Vdisks

Vdisks are created across the pdisks within a declustered array. Each recovery group requires a special *log home vdisk* to function (along with other log-type vdisks, as appropriate for specific environments); see "Log vdisks." All other vdisks are created for use as GPFS file system NSDs.

A recovery group can contain at most 64 vdisks, and the first must be the log vdisk. Vdisks can be allocated arbitrarily among declustered arrays. Vdisks are created with the **mmcrvdisk** command. The **mmdelvdisk** command destroys vdisks and all their contained data.

When creating a vdisk, you must specify the RAID code, block size, vdisk size, and a name that is unique within the recovery group and the GPFS cluster. There are no adjustable parameters available for vdisks.

RAID code

The type, performance, and space efficiency of the RAID codes used for vdisks, discussed in "RAID codes" on page 154, should be considered when choosing the RAID code for a particular set of user data. GPFS storage pools and policy-based data placement can be used to ensure data is stored with appropriate RAID codes.

Block size

The vdisk block size must equal the GPFS file system block size of the storage pool where the vdisk is assigned. For replication codes, the supported block sizes are 256 KiB, 512 KiB, 1 MiB and 2 MiB. For Reed-Solomon codes, they are 1 MiB, 2 MiB, 4 MiB, 8 MiB and 16 MiB. See "Planning considerations for GPFS Native RAID" on page 178 for an overview of vdisk configuration considerations.

Vdisk size

The minimum vdisk size is 1 GiB. The maximum vdisk size is the total space available on the pdisks in the declustered array, taking into account the overhead of the RAID code, minus spare space, minus vdisk configuration data, and minus a small amount of space reserved as a buffer for write operations. GPFS Native RAID will round up the requested vdisk size as required. When creating a vdisk, the user can specify to use all remaining space in the declustered array for that vdisk.

Log vdisks

There are four types of log vdisks:

log home vdisk
: Every recovery group requires one log home vdisk to function. The log home vdisk must be created before any other vdisks in the recovery group, and it can only be deleted after all other vdisks in the recovery group have been deleted. The log home vdisk is divided into four sublogs: long-term event log, short-term event log, metadata log, and fast-write log to log small writes.

log tip vdisk
: The log tip vdisk (appropriate for certain environments, but not required for all) is a vdisk to which log records are initially written, then migrated to the log home vdisk. The intent is to use a small, high-performance device for the log tip, and a larger vdisk on conventional spinning disks for the log home vdisk. The fast writes to the log tip hide the latency of the spinning disks used for the main body of the log.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 189

log tip backup vdisk
The log tip backup vdisk (appropriate for certain environments, but not required for all) is used as an additional replica of the log tip vdisk when the log tip vdisk is two-way replicated on nonvolatile RAM disks. Ideally, the log tip backup vdisk provides a level of performance between that of NVRAM disks and that of spinning disks.

log reserved vdisk
Log reserved vdisks are optional vdisks that are used when the log home disk is not allocated in its own declustered array. Log reserved vdisks have the same size as the log home vdisk and are used to equalize the space consumption on the data declustered arrays, but they are otherwise unused.

Typical configurations

The following are descriptions of typical vdisk configurations in various recovery group environments:

P775 Configuration
In this configuration, the log home vdisk is allocated on a declustered array made up of four SSDs. Only three-way and four-way replication codes are supported for the log home vdisk. In the typical system with four SSDs and with spare space equal to the size of one disk, the three-way replication code would be used for the log home vdisk

GPFS Storage Server with no non-volatile RAM disks
In this configuration, a three-way replicated log tip vdisk is allocated on a declustered array made up of three SSDs. A four-way replicated log home vdisk is allocated in the first declustered array of HDDs.

GPFS Storage Server with non-volatile RAM disks
In this configuration, a two-way replicated log tip vdisk is allocated on non-volatile RAM disks; one from each of the servers.

A log tip backup vdisk is allocated on a declustered array of one or more SSDs. This provides an additional copy of the log tip data when one of the NVRAM disks is unavailable. If only one SSD is used, then the log tip backup uses a raid code of **Unreplicated**.

A four-way replicated log home vdisk is allocated in the first declustered array of HDDs. A four-way replicated log reserved vdisk for each of the data declustered arrays that do not contain the log home vdisk.

The relationship between vdisks and NSDs

After creating a vdisk with the **mmcrvdisk** command, NSDs are created from the vdisks by using the **mmcrnsd** command. The relationship between vdisks and NSDs is described as follows:

- GPFS file systems are built from vdisk NSDs in the same way as they are built from any other NSDs.
- While an NSD exists for a vdisk, that vdisk cannot be deleted.
- A node cannot serve both vdisk-based NSDs and non-vdisk-based NSDs.
- A file system cannot support both vdisk-based NSDs and non-vdisk-based NSDs.
- Vdisk NSDs should not be used as tiebreaker disks.

Vdisk states

GPFS Native Raid reports vdisk states, which are displayed using the **mmlsvdisk** command and also on the **mmlsrecoverygroup** command if the **-L** option is used. Vdisks are normally in the **ok** state, which indicates that the vdisk is fully functional with full redundancy. When a pdisk failure affects a specific vdisk, the vdisk state will be reported as degraded until the affected tracks have been rebuilt onto spare space.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 190

When enough pdisks have failed that the specific vdisk has no redundancy left, the vdisk state will be reported as `critical` until the critically affected tracks have been rebuilt onto spare space. If the system uses up all the available spare space while a vdisk is in the degraded or critical state, the state will be followed by `(need spare)`, which indicates that rebuild activity is stalled, and will resume once the failed pdisks have been replaced.

Table 34. Vdisk states

State	Description
`ok`	The vdisk is functioning normally.
m`/`*n* `-degraded`	The vdisk is currently running in degraded mode: *m* is the number of pdisks that are draining *n* is the fault-tolerance of the vdisk.
`critical`	The vdisk is currently running in degraded mode and can tolerate no more pdisk losses.
`(need spare)`	Rebuild will resume when more spare space is available; applies to degraded and critical states.

Maintenance

Very large disk systems, with thousands or tens of thousands of disks and servers, will likely experience a variety of failures during normal operation. To maintain system productivity, the vast majority of these failures must be handled automatically without loss of data, without temporary loss of access to the data, and with minimal impact on the performance of the system. Some failures require human intervention, such as replacing failed components with spare parts or correcting faults that cannot be corrected by automated processes.

Disk diagnosis

The disk hospital was introduced in "Disk hospital" on page 160. When an individual disk I/O operation (read or write) encounters an error, GPFS Native RAID completes the NSD client request by reconstructing the data (for a read) or by marking the unwritten data as stale and relying on successfully written parity or replica strips (for a write), and starts the disk hospital to diagnose the disk. While the disk hospital is diagnosing, the affected disk will not be used for serving NSD client requests.

Similarly, if an I/O operation does not complete in a reasonable time period, it is timed out, and the client request is treated just like an I/O error. Again, the disk hospital will diagnose what went wrong. If the timed-out operation is a disk write, the disk remains temporarily unusable until a pending timed-out write (`PTOW`) completes.

The disk hospital then determines the exact nature of the problem. If the cause of the error was an actual media error on the disk, the disk hospital marks the offending area on disk as temporarily unusable, and overwrites it with the reconstructed data. This cures the media error on a typical HDD by relocating the data to spare sectors reserved within that HDD.

If the disk reports that it can no longer write data, the disk is marked as `readonly`. This can happen when no spare sectors are available for relocating in HDDs, or the flash memory write endurance in SSDs was reached. Similarly, if a disk reports that it cannot function at all, for example not spin up, the disk hospital marks the disk as `dead`.

The disk hospital also maintains various forms of *disk badness*, which measure accumulated errors from the disk. If the badness level is high, the disk can be marked `dead`. For less severe cases, the disk can be marked `failing`.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 191

Finally, the GPFS Native RAID server might lose communication with a disk. This can either be caused by an actual failure of an individual disk, or by a fault in the disk interconnect network. In this case, the disk is marked as `missing`.

If a disk would have to be marked `dead` or `missing`, and the problem affects only individual disks, not a large set of disks, the disk hospital attempts to recover the disk. If the disk reports that it is not started, the disk hospital attempts to start the disk. If nothing else helps, the disk hospital power-cycles the disk, and then waits for the disk to return online.

Before actually reporting an individual disk as `missing`, the disk hospital starts a search for that disk by polling all disk interfaces to locate the disk. Only after that fast poll fails is the disk actually declared `missing`.

If a large set of disks has faults, the GPFS Native RAID server can continue to serve read and write requests, provided that the number of failed disks does not exceed the fault tolerance of either the RAID code for the vdisk or the GPFS Native RAID internal configuration data. When any disk fails, the server begins rebuilding its data onto spare space. If the failure is not considered *critical*, rebuilding is throttled when user workload is present. This ensures that the performance impact to user workload is minimal. A failure might be considered critical if a vdisk has no remaining redundancy information, for example three disk faults for 4-way replication and 8 + 3p or two disk faults for 3-way replication and 8 + 2p. During a critical failure, critical rebuilding will run as fast as possible because the vdisk is in imminent danger of data loss, even if that impacts the user workload. Since the data is declustered, or spread out over many disks, and all disks in the declustered array participate in rebuilding, a vdisk will remain in critical rebuild only for short periods (2-3 minutes for a typical system). A double or triple fault is extremely rare, so the performance impact of critical rebuild is minimized.

In a multiple fault scenario, the server might not have enough disks to fulfill a request. More specifically, such a scenario occurs if the number of unavailable disks exceeds the fault tolerance of the RAID code. If some of the disks are only temporarily unavailable, and are expected back online soon, the server will stall the client I/O and wait for the disk to return to service. Disks can be temporarily unavailable for any of the following reasons:

- The disk hospital is diagnosing an I/O error.
- A timed-out write operation is pending.
- A user intentionally `suspended` the disk, perhaps it is on a carrier with another failed disk that has been removed for service.

If too many disks become unavailable for the primary server to proceed, it will fail over. In other words, the whole recovery group is moved to the backup server. If the disks are not reachable from the backup server either, then all vdisks in that recovery group become unavailable until connectivity is restored.

A vdisk will suffer data loss when the number of permanently failed disks exceeds the vdisk fault tolerance. This data loss is reported to NSD clients when the data is accessed.

Background tasks

While GPFS Native RAID primarily performs NSD client read and write operations in the foreground, it also performs several long-running maintenance tasks in the background, which are referred to as *background tasks*. The background task that is currently in progress for each declustered array is reported in the long-form output of the **mmlsrecoverygroup** command. Table 35 describes the long-running background tasks.

Table 35. Background tasks

Task	Description
`repair-RGD/VCD`	Repairing the internal recovery group data and vdisk configuration data from the failed disk onto the other disks in the declustered array.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 192

Table 35. Background tasks (continued)

Task	Description
`rebuild-critical`	Rebuilding virtual tracks that cannot tolerate any more disk failures.
`rebuild-1r`	Rebuilding virtual tracks that can tolerate only one more disk failure.
`rebuild-2r`	Rebuilding virtual tracks that can tolerate two more disk failures.
`rebuild-offline`	Rebuilding virtual tracks where failures exceeded the fault tolerance.
`rebalance`	Rebalancing the spare space in the declustered array for either a `missing` pdisk that was discovered again, or a new pdisk that was added to an existing array.
`scrub`	Scrubbing vdisks to detect any silent disk corruption or latent sector errors by reading the entire virtual track, performing checksum verification, and performing consistency checks of the data and its redundancy information. Any correctable errors found are fixed.

Server failover

If the primary GPFS Native RAID server loses connectivity to a sufficient number of disks, the recovery group attempts to fail over to the backup server. If the backup server is also unable to connect, the recovery group becomes unavailable until connectivity is restored. If the backup server had taken over, it will relinquish the recovery group to the primary server when it becomes available again.

Data checksums

GPFS Native RAID stores checksums of the data and redundancy information on all disks for each vdisk. Whenever data is read from disk or received from an NSD client, checksums are verified. If the checksum verification on a data transfer to or from an NSD client fails, the data is retransmitted. If the checksum verification fails for data read from disk, the error is treated similarly to a media error:

- The data is reconstructed from redundant data on other disks.
- The data on disk is rewritten with reconstructed good data.
- The disk badness is adjusted to reflect the silent read error.

Disk replacement

When one disk fails, the system will rebuild the data that was on the failed disk onto spare space and continue to operate normally, but at slightly reduced performance because the same workload is shared among fewer disks. With the default setting of two spare disks for each large declustered array, failure of a single disk would typically not be a sufficient reason for maintenance.

When several disks fail, the system continues to operate even if there is no more spare space. The next disk failure would make the system unable to maintain the redundancy the user requested during vdisk creation. At this point, a service request is sent to a maintenance management application that requests replacement of the failed disks and specifies the disk FRU numbers and locations.

In general, disk maintenance is requested when the number of failed disks in a declustered array reaches the disk replacement threshold. By default, that threshold is identical to the number of spare disks. For a more conservative disk replacement policy, the threshold can be set to smaller values using the **mmchrecoverygroup** command.

Disk maintenance is performed using the **mmchcarrier** command with the **--release** option, which:

- Suspends all functioning disks on the carrier that is shared with the disk being replaced.
- Powers down all the disks on that carrier.
- Turns on indicators on the disk enclosure and carrier to help locate and identify the disk that requires replacement.
- Unlocks the carrier for disk replacement.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 193

After the disk is replaced and the carrier reinserted, another **mmchcarrier** command with the **--replace** option powers on the disks.

Other hardware service

While GPFS Native RAID can easily tolerate a single disk fault with no significant impact, and failures of up to three disks with various levels of impact on performance and data availability, it still relies on the vast majority of all disks being functional and reachable from the server. If a major equipment malfunction prevents both the primary and backup server from accessing more than that number of disks, or if those disks are actually destroyed, all vdisks in the recovery group will become either unavailable or suffer permanent data loss. As GPFS Native RAID cannot recover from such catastrophic problems, it also does not attempt to diagnose them or orchestrate their maintenance.

In the case that a GPFS Native RAID server becomes permanently disabled, a manual failover procedure exists that requires recabling to an alternate server (see the **mmchrecoverygroup** command in the *GPFS: Administration and Programming Reference*). If both the primary and backup GPFS Native RAID servers for a recovery group fail, the recovery group is unavailable until one of the servers is repaired.

Component configuration in the GPFS Storage Server

The function described in this topic is only available with the IBM System x GPFS Storage Server.

GPFS uses component information in the cluster configuration data to perform configuration and validation tasks. For example, when a disk needs to be replaced, the system reports the rack location of the enclosure containing the disk along with a description of the disk's slot within the enclosure.

A component in this context refers to some resource, usually hardware, that is part of the GPFS cluster; for example, **rack** or **storage enclosure**. Each component has an associated component specification that is identified by its part number. You can use the **mmlscompspec** command to see the defined component specifications.

You will normally configure components when deploying a cluster or when adding new hardware to a cluster. The cluster must exist before you can run any of the component-related commands. The basic configuration steps are as follows:

1. Add components to the configuration using **mmdiscovercomp** and **mmaddcomp**.
2. Define the location of storage enclosures and servers using **mmchcomploc**.
3. Set the two-digit ID visible on the back of some storage enclosures using **mmsyncdisplayid**.
4. Update the component names and other attributes using **mmchcomp**.

The following commands use or change the cluster's component configuration:

mmaddcomp
: Adds new components

mmchcomp
: Change component attributes

mmchcomploc
: Change component locations

mmdelcomp
: Delete components

mmdiscovercomp
: Discover components and add them to the configuration

mmlscomp
: List components.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 194

mmlscomploc
List component locations

mmlscompspec
List component specifications

mmsyncdisplayid
Synchronize enclosure display IDs with the cluster configuration

The options that these commands support are documented in the Commands section of *GPFS: Administration and Programming Reference*. Most commands allow you to make a single change using command-line arguments. They also support stanza file input for making any number of changes with a single command.

Adding components to the cluster's configuration

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

This section discusses how to add components to the cluster's configuration. Before adding any components, you may want to view the available component specifications. To do so, you can use **mmlscompspec**, which will list the defined component specifications, identified by part number.

```
$ mmlscompspec

    Rack Specifications

Part Number  Height  Description
-----------  ------  -------------------------------------------------
1410HEA          42  Intelligent Cluster 42U 1200mm Deep Expansion Rack
1410HPA          42  Intelligent Cluster 42U 1200mm Deep Primary Rack

    Server Specifications

Part Number  Height  Description
-----------  ------  -------------------
7915AC1           2  IBM System x3650 M4

    Storage Enclosure Specifications

Part Number  Height  Description                 Vendor ID  Product ID  Drive Slots  Has Display ID
-----------  ------  --------------------------  ---------  ----------  -----------  --------------
181880E           4  DCS3700 Expansion Enclosure  IBM        DCS3700              60               1
```

If this was the first component command run on the cluster, you will see some initialization messages before the command output. The list of component specifications will be short since it only includes supported GSS hardware.

You can use **mmdiscovercomp** to define some of the cluster's components. This command finds supported storage enclosures attached to any of the cluster nodes and adds them to the cluster's configuration. (In your version of GPFS, **mmdiscovercomp** may find additional component types.) The output below resembles a cluster that includes two GSS 24 units.

```
$ mmdiscovercomp all
Adding enclosure: serial number = SV12345001; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345007; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345003; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345005; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345004; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345002; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345008; vendor ID = IBM; product ID = DCS3700
Adding enclosure: serial number = SV12345006; vendor ID = IBM; product ID = DCS3700

Storage Enclosures
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 195

```
Status  Comp ID  Component Type    Part Number  Serial Number  Name                Display ID
------  -------  ----------------  -----------  -------------  ------------------  ----------
new           1  storageEnclosure  181880E      SV12345001     181880E-SV12345001  00
new           2  storageEnclosure  181880E      SV12345007     181880E-SV12345007  00
new           3  storageEnclosure  181880E      SV12345003     181880E-SV12345003  00
new           4  storageEnclosure  181880E      SV12345005     181880E-SV12345005  00
new           5  storageEnclosure  181880E      SV12345004     181880E-SV12345004  00
new           6  storageEnclosure  181880E      SV12345002     181880E-SV12345002  00
new           7  storageEnclosure  181880E      SV12345008     181880E-SV12345008  00
new           8  storageEnclosure  181880E      SV12345006     181880E-SV12345006  00
```

In this example, **mmdiscovercomp** found eight storage enclosures that were not already in the cluster configuration. Notice that each component gets a default name. Below we will discuss how you can change the name to some identifier that is more suitable for your environment. You may run **mmdiscovercomp** a second time, in which case it should not find any new components.

Note: The serial numbers used by GPFS to identify 181880E storage enclosures do not match the product serial numbers shown on the outside of the enclosure. You may want to record this visible serial number as part of the enclosure's name. For instance, you could rename an enclosure to **G1E3-SX98760044**, which would stand for GSS 1 (G1), enclosure 3 (E3), with product serial number SX98760044. For details about how to change the name of a component using the **mmchcomp** command, see "Updating component attributes" on page 176.

The other essential components are racks to hold the storage enclosures. Use the **mmlscompspec** command to see the available rack part numbers, then use the **mmaddcomp** command to define the racks. This example shows defining an "Intelligent Cluster™ 42U 1200mm Deep Primary Rack" and giving it the name **R01C01**.

```
$ mmaddcomp 1410HPA --name R01C01
$ mmlscomp

    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
      9  1410HPA                     R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  181880E      SV12345001     181880E-SV12345001
      2  181880E      SV12345007     181880E-SV12345007
      3  181880E      SV12345003     181880E-SV12345003
      4  181880E      SV12345005     181880E-SV12345005
      5  181880E      SV12345004     181880E-SV12345004
      6  181880E      SV12345002     181880E-SV12345002
      7  181880E      SV12345008     181880E-SV12345008
      8  181880E      SV12345006     181880E-SV12345006
```

Defining component locations

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

You use the **mmchcomploc** command to place a component in a rack or change its location. This example puts the storage enclosure SV12345003 in rack R01C01 at U location 13.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 196

```
$ mmchcomploc SV12345003 R01C01 13
$ mmlscomploc
Component           Location
------------------  ------------------
181880E-SV12345003  Rack R01C01 U13-16
```

The first argument, which identifies the component, can be either the component ID, serial number, or name. The second argument is the rack (container) and the third argument is the U position in the rack.

Changing one component at a time can be slow because each change is propagated to all nodes in the cluster. You may want to use a stanza file to make multiple changes with a single command. The following example uses a stanza file to position the remaining enclosures:

```
$ cat comploc.stanza
%compLoc: compId=SV12345007 containerId=R01C01 position=1
%compLoc: compId=SV12345005 containerId=R01C01 position=5
%compLoc: compId=SV12345003 containerId=R01C01 position=13
%compLoc: compId=SV12345008 containerId=R01C01 position=17
%compLoc: compId=SV12345001 containerId=R01C01 position=21
%compLoc: compId=SV12345002 containerId=R01C01 position=25
%compLoc: compId=SV12345006 containerId=R01C01 position=33
%compLoc: compId=SV12345004 containerId=R01C01 position=37

$ mmchcomploc -F comploc.stanza
$ mmlscomploc
Component           Location
------------------  ------------------
181880E-SV12345007  Rack R01C01 U01-04
181880E-SV12345005  Rack R01C01 U05-08
181880E-SV12345003  Rack R01C01 U13-16
181880E-SV12345008  Rack R01C01 U17-20
181880E-SV12345001  Rack R01C01 U21-24
181880E-SV12345002  Rack R01C01 U25-28
181880E-SV12345006  Rack R01C01 U33-36
181880E-SV12345004  Rack R01C01 U37-40
```

Synchronizing display IDs

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

The GSS disk enclosures have a two-digit display on their ESMs. This display is visible from the back of the enclosure. When configured correctly, the two-digit display will be the same as the U location of the enclosure within its rack. **mmsyncdisplayid** sets the display on the ESMs to match the locations of the enclosure as defined by the component configuration. Here is an example:

```
$ mmsyncdisplayid all
$ mmlscomp --type storageenclosure

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  181880E      SV12345001     181880E-SV12345001  21
      2  181880E      SV12345007     181880E-SV12345007  01
      3  181880E      SV12345003     181880E-SV12345003  13
      4  181880E      SV12345005     181880E-SV12345005  05
      5  181880E      SV12345004     181880E-SV12345004  37
      6  181880E      SV12345002     181880E-SV12345002  25
      7  181880E      SV12345008     181880E-SV12345008  17
      8  181880E      SV12345006     181880E-SV12345006  33
```

With the display ID set on the actual hardware, you should now verify that the locations defined in the cluster's configuration actually match the enclosure's position in its rack. If you find that the enclosures

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 197

are not numbered 01, 05, 13, ... starting from the bottom of the rack, then use the **mmchcomploc** command to correct the configuration. Then rerun **mmsyncdisplayid** and recheck the hardware.

Updating component attributes

The function described in this topic is only available with the IBM System x GPFS Storage Server.

Note: Sample output in this topic may not match the output produced on your system.

You can change some component attributes such as the name or serial number. This example updates the name of the third enclosure so that it includes the product serial number as recommended:

```
$ mmchcomp 3 --name G1E3-SX98760044
$ mmlscomp --type storageenclosure

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      1  181880E      SV12345001     181880E-SV12345001  21
      2  181880E      SV12345007     181880E-SV12345007  01
      3  181880E      SV12345003     G1E3-SX98760044     13
      4  181880E      SV12345005     181880E-SV12345005  05
      5  181880E      SV12345004     181880E-SV12345004  37
      6  181880E      SV12345002     181880E-SV12345002  25
      7  181880E      SV12345008     181880E-SV12345008  17
      8  181880E      SV12345006     181880E-SV12345006  33
```

The **mmchcomp** *Component* argument may be either the component ID, the component serial number, or the component name. This example uses the component ID (the value 3).

Changing one component at a time can be slow, so you can use a stanza file to make multiple changes with a single command. As an aid, **mmlscomp** can list components in stanza format. You can capture this output into a file, edit the file to make the desired updates, then use the stanza file as input to **mmchcomp**. The following example uses this procedure to rename the remaining enclosures:

```
$ mmlscomp --format stanza > rename.stanza
$ cat rename.stanza
%comp:
  compId=9
  compType=rack
  partNumber=1410HEA

%comp:
  compId=1
  compType=storageEnclosure
  displayId=21
  name='181880E-SV12345001'
  partNumber=181880E
  serialNumber=SV12345001

%comp:
  compId=2
  compType=storageEnclosure
  displayId=1
  name='181880E-SV12345007'
  partNumber=181880E
  serialNumber=SV12345007

%comp:
  compId=3
  compType=storageEnclosure
  displayId=13
  name='G1E3-SX98760044'
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 198

```
  partNumber=181880E
  serialNumber=SV12345003

...
```

Edit `rename.stanza` to define new enclosure names, then apply the changes using **mmchcomp**.

```
$ vi rename.stanza
$ cat rename.stanza
%comp:
  compId=9
  compType=rack
  name=R01C01
  partNumber=1410HEA

%comp:
  compId=1
  compType=storageEnclosure
  displayId=21
  name='G1E5-SX98760048'
  partNumber=181880E
  serialNumber=SV12345001

%comp:
  compId=2
  compType=storageEnclosure
  displayId=1
  name='G1E1-SX98760044'
  partNumber=181880E
  serialNumber=SV12345007

%comp:
  compId=3
  compType=storageEnclosure
  displayId=13
  name='G1E3-SX98760041'
  partNumber=181880E
  serialNumber=SV12345003

...

$ mmchcomp -F comp.stanza
$ mmlscomp --sort name

    Rack Components

Comp ID  Part Number  Serial Number  Name
-------  -----------  -------------  ------
      9  1410HEA                     R01C01

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name                Display ID
-------  -----------  -------------  ------------------  ----------
      2  181880E      SV12345007     G1E1-SX98760044     01
      4  181880E      SV12345005     G1E2-SX98760049     05
      3  181880E      SV12345003     G1E3-SX98760041     13
      7  181880E      SV12345008     G1E4-SX98760036     17
      1  181880E      SV12345001     G1E5-SX98760048     21
      6  181880E      SV12345002     G1E6-SX98760050     25
      8  181880E      SV12345006     G1E7-SX98760039     33
      5  181880E      SV12345004     G1E8-SX98760052     37
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 199

Overall management of GPFS Native RAID

This section summarizes how to plan and monitor a GPFS Native RAID system. For an example of setting up a GPFS Native RAID system, see "GPFS Native RAID setup and disk replacement on the IBM Power 775 Disk Enclosure" on page 182.

Planning considerations for GPFS Native RAID

Planning a GPFS Native RAID implementation requires consideration of the nature of the JBOD arrays being used, the required redundancy protection and usable disk capacity, the required spare capacity and maintenance strategy, and the ultimate GPFS file system configuration. This section is a set of best-practice recommendations for using GPFS Native RAID.

- Assign a primary and backup server to each recovery group.

  Each JBOD array should be connected to two servers to protect against server failure. Each server should also have two independent paths to each physical disk to protect against path failure and provide higher throughput to the individual disks.

  Define multiple recovery groups on a JBOD array, if the architecture suggests it, and use mutually reinforcing primary and backup servers to spread the processing evenly across the servers and the JBOD array.

  Recovery group server nodes can be designated GPFS quorum or manager nodes, but they should otherwise be dedicated to GPFS Native RAID and not run application workload.

- Configure recovery group servers with a large vdisk track cache and a large pagepool.

  The **nsdRAIDTracks** configuration parameter tells GPFS Native RAID how many vdisk track descriptors, not including the actual track data, to cache in memory.

  In general, a large number of vdisk track descriptors should be cached. The **nsdRAIDTracks** value for the recovery group servers should be 10000 - 60000. If the expected vdisk NSD access pattern is random across all defined vdisks and within individual vdisks, a larger value for **nsdRAIDTracks** might be warranted. If the expected access pattern is sequential, a smaller value can be sufficient.

  The amount of actual vdisk data (including user data, parity, and checksums) that can be cached depends on the size of the GPFS pagepool on the recovery group servers and the percentage of pagepool reserved for GPFS Native RAID. The **nsdRAIDBufferPoolSizePct** parameter specifies what percentage of the pagepool should be used for vdisk data. The default is 50%, but it can be set as high as 90% or as low as 10%. Because a recovery group server is also an NSD server and the vdisk buffer pool also acts as the NSD buffer pool, the configuration parameter **nsdBufSpace** should be reduced to its minimum value of 10%.

  As an example, to have a recovery group server cache 20000 vdisk track descriptors (**nsdRAIDTracks**), where the data size of each track is 4 MiB, using 80% (**nsdRAIDBufferPoolSizePct**) of the pagepool, an approximate pagepool size of 20000 * 4 MiB * (100/80) ≈ 100000 MiB ≈ 98 GiB would be required. It is not necessary to configure the pagepool to cache all the data for every cached vdisk track descriptor, but this example calculation can provide some guidance in determining appropriate values for **nsdRAIDTracks** and **nsdRAIDBufferPoolSizePct**.

- Define each recovery group with at least one large declustered array.

  A large declustered array contains enough pdisks to store the required redundancy of GPFS Native RAID vdisk configuration data. This is defined as at least nine pdisks plus the effective spare capacity. A minimum spare capacity equivalent to two pdisks is strongly recommended in each large declustered array. The code width of the vdisks must also be considered. The effective number of non-spare pdisks must be at least as great as the largest vdisk code width. A declustered array with two effective spares where 11 is the largest code width (8 + 3p Reed-Solomon vdisks) must contain at least 13 pdisks. A declustered array with two effective spares where 10 is the largest code width (8 + 2p Reed-Solomon vdisks) must contain at least 12 pdisks.

- Place the log vdisk in a separate declustered array of solid-state disks (SSDs).

  The SSDs in the JBOD array should be used for the log vdisk of each recovery group. These SSDs should be isolated in a small log declustered array, and the log vdisk should be the only vdisk defined

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 200

there. One pdisk of spare capacity should be defined, which is the default for a small declustered array. For example, if the log declustered array contains four physical SSDs, it should have one spare defined and the log vdisk should use 3-way replication. The recommended track size for the log vdisk is 1MiB, and the recommended total size is 2 - 4 GiB.

- Determine the declustered array maintenance strategy.

  Disks will fail and need replacement, so a general strategy of deferred maintenance can be used. For example, failed pdisks in a declustered array are only replaced when the spare capacity of the declustered array is exhausted. This is implemented with the replacement threshold for the declustered array set equal to the effective spare capacity. This strategy is useful in installations with a large number of recovery groups where disk replacement might be scheduled on a weekly basis. Smaller installations can have GPFS Native RAID require disk replacement as disks fail, which means the declustered array replacement threshold can be set to one.

- Choose the vdisk RAID codes based on GPFS file system usage.

  The choice of vdisk RAID codes depends on the level of redundancy protection required versus the amount of actual space required for user data, and the ultimate intended use of the vdisk NSDs in a GPFS file system.

  Reed-Solomon vdisks are more space efficient. An 8 + 3p vdisk uses approximately 27% of actual disk space for redundancy protection and 73% for user data. An 8 + 2p vdisk uses 20% for redundancy and 80% for user data. Reed-Solomon vdisks perform best when writing whole tracks (the GPFS block size) at once. When partial tracks of a Reed-Solomon vdisk are written, parity recalculation must occur.

  Replicated vdisks are less space efficient. A vdisk with 3-way replication uses approximately 67% of actual disk space for redundancy protection and 33% for user data. A vdisk with 4-way replication uses 75% of actual disk space for redundancy and 25% for user data. The advantage of vdisks with *N*-way replication is that small or partial write operations can complete faster.

  For file system applications where write performance must be optimized, the preceding considerations make replicated vdisks most suitable for use as GPFS file system **metadataOnly** NSDs, and Reed-Solomon vdisks most suitable for use as GPFS file system **dataOnly** NSDs. The volume of GPFS file system metadata is usually small (1% - 3%) relative to file system data, so the impact of the space inefficiency of a replicated RAID code is minimized. The file system metadata is typically written in small chunks, which takes advantage of the faster small and partial write operations of the replicated RAID code. Applications are often tuned to write file system user data in whole multiples of the file system block size, which works to the strengths of the Reed-Solomon RAID codes both in terms of space efficiency and speed.

  When segregating vdisk NSDs for file system **metadataOnly** and **dataOnly** disk usage, the **metadataOnly** replicated vdisks can be created with a smaller block size and assigned to the GPFS file system storage pool. The **dataOnly** Reed-Solomon vdisks can be created with a larger block size and assigned to GPFS file system data storage pools. When using multiple storage pools, a GPFS placement policy must be installed to direct file system data to non-system storage pools.

  When write performance optimization is not important, it is acceptable to use Reed-Solomon vdisks as **dataAndMetadata** NSDs for better space efficiency.

  When assigning the failure groups to vdisk NSDs in a GPFS file system, the JBOD array should be considered the common point of failure. All vdisks within all recovery groups in a given JBOD array should be assigned the same failure group number.

Monitoring GPFS Native RAID

To monitor GPFS Native RAID during normal operation, use the **mmlsrecoverygroup**, **mmlspdisk**, and **mmpmon** commands. Pay particular attention to the GPFS Native RAID event log, which is visible using the **mmlsrecoverygroupevents** command.

Consider using GPFS Native RAID user exits to notify an automated system management tool if critical events, such as disk failures, occur during normal operation. For more information, see the **mmaddcallback** command in the *GPFS: Administration and Programming Reference*.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 201

If disk maintenance is indicated, use the **mmchcarrier** command to release the failed disk, replace the failed drive, and use the **mmchcarrier** command again to inform GPFS Native RAID that the failed disk has been replaced.

Displaying vdisk I/O statistics

To display vdisk I/O statistics, run **mmpmon** with the following command included in the input file:

vio_s [**f** [**rg** *RecoveryGroupName* [**da** *DeclusteredArrayName* [**v** *VdiskName*]]]] [**reset**]

This request returns strings containing vdisk I/O statistics as seen by that node. The values are presented as total values for the node, or they can be filtered with the **f** option. The reset option indicates that the statistics should be reset after the data is sampled.

If the **-p** option is specified when running **mmpmon**, the vdisk I/O statistics are provided in the form of keywords and values in the **vio_s** response. Table 36 lists and describes these keywords in the order in which they appear in the output.

Table 36. Keywords and descriptions of values provided in the **mmpmon vio_s** *response*

Keyword	Description
_n_	The IP address of the node responding. This is the address by which GPFS knows the node.
_nn_	The name by which GPFS knows the node.
_rc_	The reason or error code. In this case, the reply value is 0 (OK).
_t_	The current time of day in seconds (absolute seconds since Epoch (1970)).
_tu_	The microseconds part of the current time of day.
_rg_	The name of the recovery group.
_da_	The name of the declustered array.
_r_	The total number of read operations.
_sw_	The total number of short write operations.
_mw_	The total number of medium write operations.
pfw	The total number of promoted full track write operations.
ftw	The total number of full track write operations.
fuw	The total number of flushed update write operations.
fpw	The total number of flushed promoted full track write operations.
_m_	The total number of migrate operations.
_s_	The total number of scrub operations.
_l_	The total number log write operations.
_fc_	The total number of force consistency operations.
fix	The total number of buffer range fix operations.
ltr	The total number of log tip read operations.
lhr	The total number of log home read operations.
rgd	The total number of recovery group descriptor write operations.
meta	The total number metadata block write operations.

To display these statistics, use the sample script `/usr/lpp/mmfs/samples/vdisk/viostat`. The following shows the usage of the `viostat` script:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 202

```
viostat  [-F NodeFile | [--recovery-group RecoveryGroupName
                         [--declustered-array  DeclusteredArrayName
                         [--vdisk VdiskName]]]]
         [Interval [Count]]
```

For more information, see

- Chapter 9, "Monitoring GPFS I/O performance with the mmpmon command," on page 103

.

GPFS Native RAID callbacks

GPFS Native RAID introduces 12 new GPFS callbacks for events that can occur during recovery group operations. These callbacks can be installed by the system administrator using the **mmaddcallback** command.

The callbacks are provided primarily as a method for system administrators to take notice when important GPFS Native RAID events occur. For example, a GPFS administrator can use the **pdReplacePdisk** callback to send an e-mail to notify system operators that the replacement threshold for a declustered array was reached and that pdisks must be replaced. Similarly, the **preRGTakeover** callback can be used to inform system administrators of a possible server failover.

As notification methods, no real processing should occur in the callback scripts. GPFS Native RAID callbacks should not be installed for synchronous execution; the default of asynchronous callback execution should be used in all cases. Synchronous or complicated processing within a callback might delay GPFS daemon execution pathways and cause unexpected and undesired results, including loss of file system availability.

Table 37 lists the callbacks and their corresponding parameters available through the **mmaddcallback** command:

Table 37. GPFS Native RAID callbacks and parameters

Callbacks	Parameters
preRGTakeover	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
postRGTakeover	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
preRGRelinquish	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
postRGRelinquish	**myNode**, **rgName**, **rgErr**, **rgCount**, **rgReason**
rgOpenFailed	**myNode**, **rgName**, **rgErr**, **rgReason**
rgPanic	**myNode**, **rgName**, **rgErr**, **rgReason**
pdFailed	**myNode**, **rgName**, **daName**, **pdName**, **pdLocation**, **pdFru**, **pdWwn**, **pdState**
pdRecovered	**myNode**, **rgName**, **daName**, **pdName**, **pdLocation**, **pdFru**, **pdWwn**
pdReplacePdisk	**myNode**, **rgName**, **daName**, **pdName**, **pdLocation**, **pdFru**, **pdWwn**, **pdState**, **pdPriority**
pdPathDown	**myNode**, **rgName**, **daName**, **pdName**, **pdPath**, **pdLocation**, **pdFru**, **pdWwn**
daRebuildFailed	**myNode**, **rgName**, **daName**, **daRemainingRedundancy**
nsdCksumMismatch	**myNode**, **ckRole**, **ckOtherNode**, **ckNSD**, **ckReason**, **ckStartSector**, **ckDataLen**, **ckErrorCountClient**, **ckErrorCountServer**, **ckErrorCountNSD**, **ckReportingInterval**

All GPFS Native RAID callbacks are local, which means that the event triggering the callback occurs only on the involved node or nodes, in the case of **nsdCksumMismatch**, rather than on every node in the GPFS cluster. The nodes where GPFS Native RAID callbacks should be installed are, by definition, the recovery group server nodes. An exception is the case of **nsdCksumMismatch**, where it makes sense to install the callback on GPFS client nodes as well as recovery group servers.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 203

For more information about GPFS Native RAID callbacks, see the *GPFS: Administration and Programming Reference* topic "**mmaddcallback** command".

GPFS Native RAID setup and disk replacement on the IBM Power 775 Disk Enclosure

Example scenario: Configuring GPFS Native RAID recovery groups

This topic provides a detailed example of configuring GPFS Native RAID using the JBOD SAS disks on the Power 775 Disk Enclosure. The example considers one fully populated Power 775 Disk Enclosure cabled to two recovery group servers, and shows how the architecture of the Power 775 Disk Enclosure determines the structure of the recovery groups. Throughout this topic, it may be helpful to have Power 775 Disk Enclosure documentation at hand.

Preparing recovery group servers

Disk enclosure and HBA cabling

The Power 775 Disk Enclosure should be cabled to the intended recovery group servers according to the Power 775 Disk Enclosure hardware installation instructions. The fully populated Power 775 Disk Enclosure consists of 8 STORs of 48 disks, for a total of 384 JBOD disks. Each STOR provides redundant left and right port cards for host server HBA connections (STOR is short for *physical storage group*, meaning the part of the disk enclosure controlled by a pair of port cards). To ensure proper multi-pathing and redundancy, each recovery group server must be connected to each port card using different HBAs. For example, STOR 1 has port cards P1-C4 and P1-C5. Server 1 may be connected to P1-C4 using HBA `hba1` and to P1-C5 using HBA `hba2`; similarly for server 2 and its respective HBAs `hba1` and `hba2`.

GPFS Native RAID provides system administration tools for verifying the correct connectivity of the Power 775 Disk Enclosure, which will be seen later during the operating system preparation.

When the port cards of the Power 775 Disk Enclosure have been cabled to the appropriate HBAs of the two recovery group servers, the Power 775 Disk Enclosure should be powered on and the recovery group servers should be rebooted.

Initial operating system verification

Preparation then continues with the operating system, which must be either AIX 7.1 or Red Hat Enterprise Linux 6.1, and which must be the same on both recovery group servers. It is not necessary to do a complete verification of the Power 775 Disk Enclosure connectivity at this point. Logging in to the servers to perform a quick check that at least some disks have been detected and configured by the operating system will suffice. The operating system device configuration should be examined for the Power 775 Disk Enclosure VPD enclosure type, which is 78AD.001.

One way to quickly verify that AIX has configured devices with enclosure type 78AD.001 for the Power 775 Disk Enclosure is:

```
# lsdev -t ses -F 'name physloc parent' | grep 78AD.001
```

The output should include lines resembling the following:

```
ses12 U78AD.001.000DE37-P1-C4 sas3
```

This is the SAS expander device on port card P1-C4 of the Power 775 Disk Enclosure with serial number 000DE37, together with the SAS protocol device driver `sas3` under which it has been configured. To see what disks have been detected by the SAS protocol driver, use:

```
# lsdev -p sas3
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 204

The output should include all the disks and port card expanders that successfully configured under the sas3 SAS protocol driver (which corresponds to the HBA device mpt2sas3).

If AIX has not configured any port card expanders of enclosure type 78AD.001, the hardware installation of the server HBAs and the Power 775 Disk Enclosure must be reexamined and corrected.

One way to quickly verify that Red Hat Enterprise Linux has configured devices with enclosure type 78AD.001 for the Power 775 Disk Enclosure is:

```
# grep 78AD.001 /proc/scsi/scsi
```

The output should include lines resembling the following:

```
Vendor: IBM      Model: 78AD-001         Rev: 0150

Further examination of the /proc/scsi/scsi file should reveal contents similar to:
Host: scsi7 Channel: 00 Id: 394 Lun: 00
  Vendor: IBM      Model: ST9600204SS      Rev: 631C
  Type:   Direct-Access                    ANSI  SCSI revision: 06
Host: scsi7 Channel: 00 Id: 395 Lun: 00
  Vendor: IBM      Model: 78AD-001         Rev: 0150
  Type:   Enclosure                        ANSI  SCSI revision: 04
```

The above indicates that a Power 775 Disk Enclosure port card SAS expander and a disk drive have been configured on SCSI host bus 7 (the HBA corresponding to scsi7).

As with AIX, if Linux has not configured any port card expanders of enclosure type 78AD.001, the hardware installation of the server HBAs and the Power 775 Disk Enclosure must be reexamined and corrected.

Disabling operating system multi-pathing

Once it has been verified that at least some of the Power 775 Disk Enclosure has been configured by the operating system, the next step is to disable any operating system multi-pathing. Since GPFS Native RAID performs its own disk multi-pathing, AIX MPIO (Multiple Path I/O) and Linux DMM (Device Mapper Multipath) must be disabled as appropriate.

To disable AIX MPIO for SAS disks, use:

```
# manage_disk_drivers -d SAS_SCSD -o AIX_non_MPIO
```

To disable Linux DMM, use:

```
# chkconfig --del multipathd
```

Note: This blanket disabling of operating system multi-pathing is appropriate because a Power 775 Disk Enclosure installation provides the only available disk devices to the recovery group servers. Once operating system multi-pathing has been disabled, the recovery group servers should be rebooted.

Operating system device attributes

For best performance, the operating system disk device driver should be configured to allow GPFS Native RAID I/O operations to be made with one disk access, rather than being fragmented. Under AIX this is controlled by the **max_transfer** attribute of the HBAs and disk devices. Under Red Hat Enterprise Linux, this is controlled by the disk block device **max_sectors_kb** attribute.

The disk I/O size performed by GPFS Native RAID depends on the strip size of the RAID code of the vdisk NSD. This in turn is related to the vdisk track size and its corresponding GPFS file system block size. The operating system I/O size should be equal to or greater than the largest strip size of the planned vdisk NSDs.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 205

Because GPFS Native RAID stores checksums with each strip, strips have an additional 4 KiB or 8 KiB than might be expected just from the user data (strips containing 2 MiB of user data have an additional 8 KiB; all smaller strips have an additional 4 KiB). The strip size for a replicated vdisk RAID code is equal to the vdisk track size plus the size of the checksum. The strip size for a Reed-Solomon vdisk RAID code is equal to one-eighth of the vdisk track size plus the size of the checksum.

The default **max_transfer** value of 1 MiB under AIX is suitable for GPFS Native RAID vdisk strip sizes under 1 MiB.

The default **max_sectors_kb** value of 512 KiB sectors under Red Hat Enterprise Linux is suitable for GPFS Native RAID vdisk strip sizes under 512 KiB.

However, for vdisk strip sizes greater than 1 MiB under AIX or greater than 512 KiB under Linux, the operating system disk device driver I/O size should be increased for best performance.

The following table indicates the relationship between file system NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and the non-default operating system I/O size for all permitted GPFS Native RAID vdisks. The AIX **max_transfer** attribute is specified in hexadecimal, and the only allowable values greater than the 1 MiB default are 0x200000 (2 MiB) and 0x400000 (4 MiB). Linux **max_sectors_kb** is specified as the desired I/O size in KiB.

Table 38. NSD block size, vdisk track size, vdisk RAID code, vdisk strip size, and non-default operating system I/O size for permitted GPFS Native RAID vdisks

NSD block size	vdisk track size	vdisk RAID code	RAID code strip size	AIX max_transfer	Linux max_sectors_kb
256 KiB	256 KiB	3- or 4-way replication	260 KiB	default	260
512 KiB	512 KiB	3- or 4-way replication	516 KiB	default	516
1 MiB	1 MiB	3- or 4-way replication	1028 KiB	0x200000	1028
2 MiB	2 MiB	3- or 4-way replication	2056 KiB	0x400000	2056
512 KiB	512 KiB	8 + 2p or 8 + 3p	68 KiB	default	default
1 MiB	1 MiB	8 + 2p or 8 + 3p	132 KiB	default	default
2 MiB	2 MiB	8 + 2p or 8 + 3p	260 KiB	default	260
4 MiB	4 MiB	8 + 2p or 8 + 3p	516 KiB	default	516
8 MiB	8 MiB	8 + 2p or 8 + 3p	1028 KiB	0x200000	1028
16 MiB	16 MiB	8 + 2p or 8 + 3p	2056 KiB	0x400000	2056

If the largest strip size of all the vdisk NSDs planned for a GPFS Native RAID installation exceeds the operating system default I/O size, the operating system I/O size should be changed.

AIX Under AIX, this involves changing the HBA and disk device **max_transfer** size. For an HBA to accommodate an increased **max_transfer** size, the **max_commands** for the HBA will also need to be decreased. With a 0x400000 **max_transfer** size, the four-port HBA requires a **max_commands** value of 124 and the two-port HBA requires a **max_commands** value of 248.

To change the **max_transfer** attribute to 4 MiB for the HBA `mpt2sas0` under AIX, use the following command:

```
# chdev -P -l mpt2sas0 -a max_transfer=0x400000
```

To change the **max_commands** value to 124 for the four-port HBA `mpt2sas0` (AIX device type 001072001410f60), use the following command:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 206

```
# chdev -P -l mpt2sas0 -a max_commands=124
```

To change the **max_commands** value to 248 for the two-port HBA `mpt2sas0` (AIX device type 001072001410ea0), use the following command:

```
# chdev -P -l mpt2sas0 -a max_commands=248
```

Repeat the previous commands for each HBA.

Changing the hdisk **max_transfer** attribute requires changing its default value for AIX device type nonmpioscsd disks. It is not sufficient to change the **max_transfer** for individual hdisks, because performing disk replacement deletes and recreates hdisk objects.

To change the default hdisk **max_transfer** attribute for type nonmpioscsd hdisks, use the following command:

```
# chdef -a max_transfer=0x400000 -c disk -s sas -t nonmpioscsd
```

The new **max_transfer** and **max_commands** values will not take effect until AIX reconfigures the HBAs and hdisks. This can be done either by rebooting the recovery group server, or by deconfiguring (but not removing) and then reconfiguring the affected HBAs. The **max_transfer** and **max_commands** attribute are recorded in the CuAt ODM class and will persist across reboots.

Linux Under Linux, the **max_sectors_kb** I/O size is reset to the default each time a disk block device is configured. This means that upon reboot and upon adding or replacing disks, a desired nondefault I/O size must be explicitly set. Since **max_sectors_kb** is dynamically reconfigurable, GPFS Native RAID manages this setting under Linux.

The value of **max_sectors_kb** under Linux is set on all the disks in a recovery group when GPFS Native RAID begins serving the recovery group, and on disks that are configured as part of GPFS Native RAID disk replacement.

The default **max_sectors_kb** set by GPFS Native RAID is 4096, which is large enough for any of the strip sizes listed in Table 38 on page 184. It can be changed to exactly match the largest strip size in use by GPFS Native RAID by setting the GPFS **nsdRAIDBlockDeviceMaxSectorsKB** configuration parameter.

To set the **max_sectors_kb** value used by GPFS Native RAID to 2056, use the following command:

```
# mmchconfig nsdRAIDBlockDeviceMaxSectorsKB=2056
```

The new value will take effect the next time GPFS is started. To have the new value take effect immediately, append the **-i** option to the above command.

For optimal performance, additional device attributes may need to be changed (for example, the HBA and block device command queue depths); consult the operating system documentation for the device attributes.

Verifying that a Power 775 Disk Enclosure is configured correctly

Once a superficial inspection indicates that the Power 775 Disk Enclosure has been configured on the recovery group servers, and especially once operating system multi-pathing has been disabled, it is necessary to perform a thorough discovery of the disk topology on each server.

To proceed, GPFS must be installed on the recovery group servers, and they should be members of the same GPFS cluster. Consult the *GPFS: Administration and Programming Reference* for instructions for creating a GPFS cluster.

GPFS Native RAID provides tools in `/usr/lpp/mmfs/samples/vdisk` for collecting and collating information on any attached Power 775 Disk Enclosure and for verifying that the detected topology is correct. The **mmgetpdisktopology** command examines the operating system's list of connected devices and produces a colon-delimited database with a line for each discovered Power 775 Disk Enclosure

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 207

physical disk, port card expander device, and HBA. **mmgetpdisktopology** should be run on each of the two intended recovery group server nodes, and the results examined to verify that the disk enclosure hardware and software configuration is as expected. An additional tool called **topsummary** concisely summarizes the output of the **mmgetpdisktopology** command.

Create a directory in which to work, and then capture the output of the **mmgetpdisktopology** command from each of the two intended recovery group server nodes:

```
# mkdir p7ihde
# cd p7ihde
# ssh server1 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server1.top
# ssh server2 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server2.top
```

Then view the summary for each of the nodes (server1 example shown):

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
P7IH-DE enclosures found: DE00022
Enclosure DE00022:
Enclosure DE00022 STOR P1-C4/P1-C5 sees both portcards: P1-C4 P1-C5
Portcard P1-C4: ses0[0150]/mpt2sas0/24 diskset "37993" ses1[0150]/mpt2sas0/24 diskset "18793"
Portcard P1-C5: ses4[0150]/mpt2sas1/24 diskset "37993" ses5[0150]/mpt2sas1/24 diskset "18793"
Enclosure DE00022 STOR P1-C4/P1-C5 sees 48 disks
Enclosure DE00022 STOR P1-C12/P1-C13 sees both portcards: P1-C12 P1-C13
Portcard P1-C12: ses8[0150]/mpt2sas2/24 diskset "40657" ses9[0150]/mpt2sas2/24 diskset "44382"
Portcard P1-C13: ses12[0150]/mpt2sas3/24 diskset "40657" ses13[0150]/mpt2sas3/24 diskset "44382"
Enclosure DE00022 STOR P1-C12/P1-C13 sees 48 disks
Enclosure DE00022 STOR P1-C20/P1-C21 sees both portcards: P1-C20 P1-C21
Portcard P1-C20: ses16[0150]/mpt2sas4/24 diskset "04091" ses17[0150]/mpt2sas4/24 diskset "31579"
Portcard P1-C21: ses20[0150]/mpt2sas5/24 diskset "04091" ses21[0150]/mpt2sas5/24 diskset "31579"
Enclosure DE00022 STOR P1-C20/P1-C21 sees 48 disks
Enclosure DE00022 STOR P1-C28/P1-C29 sees both portcards: P1-C28 P1-C29
Portcard P1-C28: ses24[0150]/mpt2sas6/24 diskset "64504" ses25[0150]/mpt2sas6/24 diskset "62361"
Portcard P1-C29: ses28[0150]/mpt2sas7/24 diskset "64504" ses29[0150]/mpt2sas7/24 diskset "62361"
Enclosure DE00022 STOR P1-C28/P1-C29 sees 48 disks
Enclosure DE00022 STOR P1-C60/P1-C61 sees both portcards: P1-C60 P1-C61
Portcard P1-C60: ses30[0150]/mpt2sas7/24 diskset "10913" ses31[0150]/mpt2sas7/24 diskset "52799"
Portcard P1-C61: ses26[0150]/mpt2sas6/24 diskset "10913" ses27[0150]/mpt2sas6/24 diskset "52799"
Enclosure DE00022 STOR P1-C60/P1-C61 sees 48 disks
Enclosure DE00022 STOR P1-C68/P1-C69 sees both portcards: P1-C68 P1-C69
Portcard P1-C68: ses22[0150]/mpt2sas5/24 diskset "50112" ses23[0150]/mpt2sas5/24 diskset "63400"
Portcard P1-C69: ses18[0150]/mpt2sas4/24 diskset "50112" ses19[0150]/mpt2sas4/24 diskset "63400"
Enclosure DE00022 STOR P1-C68/P1-C69 sees 48 disks
Enclosure DE00022 STOR P1-C76/P1-C77 sees both portcards: P1-C76 P1-C77
Portcard P1-C76: ses14[0150]/mpt2sas3/23 diskset "45948" ses15[0150]/mpt2sas3/24 diskset "50856"
Portcard P1-C77: ses10[0150]/mpt2sas2/24 diskset "37258" ses11[0150]/mpt2sas2/24 diskset "50856"
Enclosure DE00022 STOR P1-C76/P1-C77 sees 48 disks
Enclosure DE00022 STOR P1-C84/P1-C85 sees both portcards: P1-C84 P1-C85
Portcard P1-C84: ses6[0150]/mpt2sas1/24 diskset "13325" ses7[0150]/mpt2sas1/24 diskset "10443"
Portcard P1-C85: ses2[0150]/mpt2sas0/24 diskset "13325" ses3[0150]/mpt2sas0/24 diskset "10443"
Enclosure DE00022 STOR P1-C84/P1-C85 sees 48 disks
Carrier location P1-C79-D4 appears only on the portcard P1-C77 path
Enclosure DE00022 sees 384 disks

mpt2sas7[1005470001] U78A9.001.9998884-P1-C1 DE00022 STOR 4 P1-C29 (ses28 ses29) STOR 5 P1-C60 (ses30 ses31)
mpt2sas6[1005470001] U78A9.001.9998884-P1-C3 DE00022 STOR 4 P1-C28 (ses24 ses25) STOR 5 P1-C61 (ses26 ses27)
mpt2sas5[1005470001] U78A9.001.9998884-P1-C5 DE00022 STOR 3 P1-C21 (ses20 ses22) STOR 6 P1-C68 (ses21 ses23)
mpt2sas4[1005470001] U78A9.001.9998884-P1-C7 DE00022 STOR 3 P1-C20 (ses16 ses17) STOR 6 P1-C69 (ses18 ses19)
mpt2sas3[1005470001] U78A9.001.9998884-P1-C9 DE00022 STOR 2 P1-C13 (ses12 ses13) STOR 7 P1-C76 (ses14 ses15)
mpt2sas2[1005470001] U78A9.001.9998884-P1-C11 DE00022 STOR 2 P1-C12 (ses8 ses9) STOR 7 P1-C77 (ses10 ses11)
mpt2sas1[1005470001] U78A9.001.9998884-P1-C13 DE00022 STOR 1 P1-C5 (ses4 ses5) STOR 8 P1-C84 (ses6 ses7)
mpt2sas0[1005470001] U78A9.001.9998884-P1-C15 DE00022 STOR 1 P1-C4 (ses0 ses1) STOR 8 P1-C85 (ses2 ses3)
```

In the preceding output, the Power 775 Disk Enclosure with serial number DE00022 is discovered, together with its eight individual STORs and the component port cards, port card expanders (with their firmware levels in brackets), and physical disks. One minor discrepancy is noted: The physical disk in

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 208

location P1-C79-D4 is only seen over one of the two expected HBA paths. This can also be seen in the output for the STOR with port cards P1-C76 and P1-C77:

```
Enclosure DE00022 STOR P1-C76/P1-C77 sees both portcards: P1-C76 P1-C77
Portcard P1-C76: ses14[0150]/mpt2sas3/23 diskset "45948" ses15[0150]/mpt2sas3/24 diskset "50856"
Portcard P1-C77: ses10[0150]/mpt2sas2/24 diskset "37258" ses11[0150]/mpt2sas2/24 diskset "50856"
Enclosure DE00022 STOR P1-C76/P1-C77 sees 48 disks
```

Here the connection through port card P1-C76 sees just 23 disks on the expander `ses14` and all 24 disks on the expander `ses15`, while the connection through port card P1-C77 sees all 24 disks on each of the expanders `ses10` and `ses11`. The "disksets" that are reached over the expanders are identified by a checksum of the unique SCSI WWNs of the physical disks that are present; equal disksets represent the same collection of physical disks.

The preceding discrepancy can either be corrected or ignored, as it is probably due to a poorly seated or defective port on the physical disk. The disk is still present on the other port.

If other discrepancies are noted (for example, physical disks that are expected but do not show up at all, or SSDs or HDDs in the wrong locations), they should be corrected before proceeding.

The HBAs (firmware levels in brackets) are also listed with their slot location codes to show the cabling pattern. Each HBA sees two STORs, and each STOR is seen by two different HBAs, which provides the multiple paths and redundancy required by a correct Power 775 Disk Enclosure installation.

This output can be compared to the hardware cabling specification to verify that the disk enclosure is connected correctly.

The `server2.top` topology database should also be examined with the **topsummary** sample script and verified to be correct.

Once the Power 775 Disk Enclosure topologies are verified to be correct on both intended recovery group server nodes, the recommended recovery group configuration can be created using GPFS Native RAID commands.

Creating recovery groups on a Power 775 Disk Enclosure

Configuring GPFS nodes to be recovery group servers

Before a GPFS node can create and serve recovery groups, it must be configured with a vdisk track cache. This is accomplished by setting the **nsdRAIDTracks** configuration parameter.

nsdRAIDTracks is the GPFS configuration parameter essential to define a GPFS cluster node as a recovery group server. It specifies the number of vdisk tracks of which the attributes will be held in memory by the GPFS daemon on the recovery group server.

The actual contents of the vdisk tracks, the user data and the checksums, are stored in the standard GPFS pagepool. Therefore, the size of the GPFS pagepool configured on a recovery group server should be considerable, on the order of tens of gigabytes. The amount of pagepool dedicated to hold vdisk track data is governed by the **nsdRAIDBufferPoolSizePct** parameter, which defaults to 50%. In practice, a recovery group server will not need to use the GPFS pagepool for any significant amount of standard file caching, and the **nsdRAIDBufferPoolSizePct** value can be increased to 80%. Also applicable, since a recovery group server is by definition an NSD server, is the **nsdBufSpace** parameter, which defaults to 30% of pagepool. Since the vdisk buffer pool doubles as the NSD buffer spool, the **nsdBufSpace** parameter should be decreased to its minimum of 10%. Together these values leave only 10% of the pagepool for application program file cache, but this should not be a problem as a recovery group server should not be running application programs.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 209

In this example, the recovery group servers will be configured to cache the information on 16384 vdisk tracks and to have 64 GiB of pagepool, of which 80% will be used for vdisk data. Once the configuration changes are made, the servers will need to be restarted.

```
# mmchconfig nsdRAIDTracks=16384,nsdRAIDBufferPoolSizePct=80,nsdBufSpace=10,pagepool=64G -N server1,server2
# mmshutdown -N server1,server2
# mmstartup -N server1,server2
```

Defining the recovery group layout

The definition of recovery groups on a Power 775 Disk Enclosure is dictated by the architecture and cabling of the disk enclosure. Two servers sharing a Power 775 Disk Enclosure implies two recovery groups; one is served by one node and one by the other, and each server acts as the other's backup. Half the disks in each STOR should belong to one recovery group, and half to the other. One recovery group will therefore be defined on the disks and carriers in the top halves of the eight STORs, and one on the bottom halves. Since the disks in a STOR are placed four to a removable carrier, thereby having a common point of failure, each disk in a carrier should belong to one of four different declustered arrays. Should a carrier fail or be removed, then each declustered array will only suffer the loss of one disk. There are four SSDs distributed among the top set of carriers, and four in the bottom set of carriers. These groups of four SSDs will make up the vdisk log declustered arrays in their respective halves.

GPFS Native RAID provides a tool that understands the layout of the Power 775 Disk Enclosure and will automatically generate the **mmcrrecoverygroup** stanza files for creating the top and bottom recovery groups. **/usr/lpp/mmfs/samples/vdisk/mkp7rginput**, when supplied with output of the **mmgetpdisktopology** command, will create recovery group stanza files for the top and bottom halves of each Power 775 Disk Enclosure found in the topology.

Each recovery group server, though it may see the same functional disk enclosure topology, will almost certainly differ in the particulars of which disk device names (e.g., `/dev/rhdisk77` on AIX or `/dev/sdax` on Linux) refer to which physical disks in what disk enclosure location.

There are two possibilities then for creating the recovery group stanza files and the recovery groups themselves:

Alternative 1:
: Generate the recovery group stanza files and create the recovery groups from the perspective of just one of the servers as if that server were to be primary for both recovery groups, and then use the **mmchrecoverygroup** command to swap the primary and backup servers for one of the recovery groups

Alternative 2:
: Generate the recovery group stanza files for each server's primary recovery group using the primary server's topology file.

This example will show both alternatives.

Creating the recovery groups, alternative 1
: To create the recovery group input stanza files from the perspective of `server1`, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server1.top
```

: This will create two files for each disk enclosure present in the `server1` topology; in this case, `DE00022TOP.server1` for the top half of disk enclosure DE00022 and `DE00022BOT.server2` for the bottom half. (An extra file, `DEXXXXXbad`, may be created if any discrepancies are present in the topology; if such a file is created by **mkp7rginput**, it should be examined and the discrepancies corrected.)

: The recovery group stanza files will follow the recommended best practice for the Power 775 Disk Enclosure of defining in each half of the disk enclosure a separate declustered array of 4

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 210

SSDs for recovery group transaction logging, and four file system data declustered arrays using the regular HDDs according to which of the four disk enclosure carrier slots each HDD resides in.

The defaults are accepted for other recovery group declustered array parameters such as scrub duration, spare space, and disk replacement policy.

The stanza file will look something like this:

```
# head DE00022TOP.server1
%pdisk: pdiskName=c081d1
        device=/dev/hdisk10
        da=DA1
%pdisk: pdiskName=c065d1
        device=/dev/hdisk211
        da=DA1
%pdisk: pdiskName=c066d1
        device=/dev/hdisk259
        da=DA1
%pdisk: pdiskName=c067d1
```

All the pdisk stanzas for declustered array DA1 will be listed first, followed by those for DA2, DA3, DA4, and the LOG declustered array. The pdisk names will indicate the carrier and disk location in which the physical disk resides. Notice that only one block device path to the disk is given; the second path will be discovered automatically soon after the recovery group is created.

Now that the DE00022TOP.server1 and DE00022BOT.server1 stanza files have been created from the perspective of recovery group server node server1, these two recovery groups can be created using two separate invocations of the **mmcrrecoverygroup** command:

```
# mmcrrecoverygroup DE00022TOP -F DE00022TOP.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmcrrecoverygroup DE00022BOT -F DE00022BOT.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Note that both recovery groups were created with server1 as primary and server2 as backup. It is now necessary to swap the primary and backup servers for DE00022BOT using the **mmchrecoverygroup** command:

```
# mmchrecoverygroup DE00022BOT --servers server2,server1
mmchrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

GPFS Native RAID will automatically discover the appropriate disk devices on server2.

Creating the recovery groups, alternative 2

To create the recovery groups from the start with the intended primary and backup servers, the stanza files from both server topologies will need to be created.

To create the server1 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server1.top
```

To create the server2 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkp7rginput server2.top
```

These two commands will result in four stanza files: DE00022TOP.server1, DE00022BOT.server1, DE00022TOP.server2, and DE00022BOT.server2. (As in alternative 1, if any files named DEXXXXXbad are created, they should be examined and the errors within should be corrected.)

The DE00022TOP recovery group must then be created using server1 as the primary and the DE00022TOP.server1 stanza file. The DE00022BOT recovery group must be created using server2 as the primary and the DE00022BOT.server2 stanza file.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 211

```
# mmcrrecoverygroup DE00022TOP -F DE00022TOP.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmcrrecoverygroup DE00022BOT -F DE00022BOT.server2 --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Since each recovery group was created using the intended primary server and the stanza file for that server, it is not necessary to swap the primary and backup servers.

Verifying recovery group creation

Use the **mmlsrecoverygroup** command to verify that each recovery group was created:

```
# mmlsrecoverygroup DE00022TOP -L

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 DE00022TOP                   5       0     192

 declustered   needs                              replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 DA2          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 DA3          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 DA4          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 LOG          no            0       4       1          1     558 GiB   14 days  inactive     0%  low

                                       declustered
 vdisk               RAID code            array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------

 active recovery group server                      servers
 ------------------------------------------------  -------
 server1                                           server1,server2

# mmlsrecoverygroup DE00022BOT -L

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 DE00022BOT                   5       0     192

 declustered   needs                              replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 DA2          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 DA3          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 DA4          no            0      47       2          2      24 TiB   14 days  inactive     0%  low
 LOG          no            0       4       1          1     558 GiB   14 days  inactive     0%  low

                                       declustered
 vdisk               RAID code            array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------

 active recovery group server                      servers
 ------------------------------------------------  -------
 server1                                           server2,server1
```

Notice that the vdisk sections of the newly created recovery groups are empty; the next step is to create the vdisks.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 212

Defining and creating the vdisks

Once the recovery groups are created and being served by their respective servers, it is time to create the vdisks using the **mmcrvdisk** command.

Each recovery group requires a single log vdisk for recording RAID updates and diagnostic information. This is internal to the recovery group, cannot be used for user data, and should be the only vdisk in the `LOG` declustered array. The log vdisks in this example use 3-way replication in order to fit in the `LOG` declustered array, which contains 4 SSDs and spare space equivalent to one disk.

Data vdisks are required to be defined in the four data declustered arrays for use as file system NSDs. In this example, each of the declustered arrays for file system data is divided into two vdisks with different characteristics: one using 4-way replication and a 1 MiB block size and a total vdisk size of 250 GiB suitable for file system metadata, and one using Reed-Solomon 8 + 3p encoding and an 16 MiB block size suitable for file system data. The vdisk size is omitted for the Reed-Solomon vdisks, meaning that they will default to use the remaining non-spare space in the declustered array (for this to work, any vdisks with specified total sizes must of course be defined first).

The possibilities for the vdisk creation stanza file are quite great, depending on the number and type of vdisk NSDs required for the number and type of file systems desired, so the vdisk stanza file will need to be created by hand, possibly following a template.

In this example, a single stanza file, `mmcrvdisk.DE00022ALL`, is used. The single file contains the specifications for all the vdisks in both the `DE00022TOP` and `DE00022BOT` recovery groups. Here is what the example stanza file for use with **mmcrvdisk** should look like:

```
# cat mmcrvdisk.DE00022ALL
%vdisk: vdiskName=DE00022TOPLOG
        rg=DE00022TOP
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=DE00022BOTLOG
        rg=DE00022BOT
        da=LOG
        blocksize=1m
        size=4g
        raidCode=3WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=DE00022TOPDA1META
        rg=DE00022TOP
        da=DA1
        blocksize=1m
        size=250g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=22
        pool=system
%vdisk: vdiskName=DE00022TOPDA1DATA
        rg=DE00022TOP
        da=DA1
        blocksize=16m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=22
        pool=data
%vdisk: vdiskName=DE00022BOTDA1META
        rg=DE00022BOT
        da=DA1
        blocksize=1m
        size=250g
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 213

```
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=22
        pool=system
%vdisk: vdiskName=DE00022BOTDA1DATA
        rg=DE00022BOT
        da=DA1
        blocksize=16m
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=22
        pool=data
[DA2, DA3, DA4 vdisks omitted.]
```

Notice how the file system metadata vdisks are flagged for eventual file system usage as **metadataOnly** and for placement in the `system` storage pool, and the file system data vdisks are flagged for eventual **dataOnly** usage in the `data` storage pool. (After the file system is created, a policy will be required to allocate file system data to the correct storage pools; see "Creating the GPFS file system" on page 193.)

Importantly, also notice that block sizes for the file system metadata and file system data vdisks must be specified at this time, may not later be changed, and must match the block sizes supplied to the eventual **mmcrfs** command.

Notice also that the eventual `failureGroup=22` value for the NSDs on the file system vdisks is the same for vdisks in both the `DE00022TOP` and `DE00022BOT` recovery groups. This is because the recovery groups, although they have different servers, still share a common point of failure in the disk enclosure `DE00022`, and GPFS should be informed of this through a distinct failure group designation for each disk enclosure. It is up to the GPFS system administrator to decide upon the failure group numbers for each Power 775 Disk Enclosure in the GPFS cluster.

To create the vdisks specified in the `mmcrvdisk.DE00022ALL` file, use the following **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisk.DE00022ALL
mmcrvdisk: [I] Processing vdisk DE00022TOPLOG
mmcrvdisk: [I] Processing vdisk DE00022BOTLOG
mmcrvdisk: [I] Processing vdisk DE00022TOPDA1META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA1DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA2META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA2DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA3META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA3DATA
mmcrvdisk: [I] Processing vdisk DE00022TOPDA4META
mmcrvdisk: [I] Processing vdisk DE00022TOPDA4DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA1META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA1DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA2META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA2DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA3META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA3DATA
mmcrvdisk: [I] Processing vdisk DE00022BOTDA4META
mmcrvdisk: [I] Processing vdisk DE00022BOTDA4DATA
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Creation of the vdisks may be verified through the **mmlsvdisk** command (the **mmlsrecoverygroup** command may also be used):

```
# mmlsvdisk
                                                  declustered  block size
 vdisk name          RAID code        recovery group         array       in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 DE00022BOTDA1DATA   8+3p             DE00022BOT          DA1               16384
 DE00022BOTDA1META   4WayReplication  DE00022BOT          DA1                1024
 DE00022BOTDA2DATA   8+3p             DE00022BOT          DA2               16384
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 214

```
 DE00022BOTDA2META   4WayReplication  DE00022BOT          DA2               1024
 DE00022BOTDA3DATA   8+3p             DE00022BOT          DA3              16384
 DE00022BOTDA3META   4WayReplication  DE00022BOT          DA3               1024
 DE00022BOTDA4DATA   8+3p             DE00022BOT          DA4              16384
 DE00022BOTDA4META   4WayReplication  DE00022BOT          DA4               1024
 DE00022BOTLOG       3WayReplication  DE00022BOT          LOG               1024  log
 DE00022TOPDA1DATA   8+3p             DE00022TOP          DA1              16384
 DE00022TOPDA1META   4WayReplication  DE00022TOP          DA1               1024
 DE00022TOPDA2DATA   8+3p             DE00022TOP          DA2              16384
 DE00022TOPDA2META   4WayReplication  DE00022TOP          DA2               1024
 DE00022TOPDA3DATA   8+3p             DE00022TOP          DA3              16384
 DE00022TOPDA3META   4WayReplication  DE00022TOP          DA3               1024
 DE00022TOPDA4DATA   8+3p             DE00022TOP          DA4              16384
 DE00022TOPDA4META   4WayReplication  DE00022TOP          DA4               1024
 DE00022TOPLOG       3WayReplication  DE00022TOP          LOG               1024  log
```

Creating NSDs from vdisks

The **mmcrvdisk** command rewrites the input file so that it is ready to be passed to the **mmcrnsd** command that creates the NSDs from which GPFS builds file systems. To create the vdisk NSDs, run the **mmcrnsd** command on the rewritten **mmcrvdisk** stanza file:

```
# mmcrnsd -F mmcrvdisk.DE00022ALL
mmcrnsd: Processing disk DE00022TOPDA1META
mmcrnsd: Processing disk DE00022TOPDA1DATA
mmcrnsd: Processing disk DE00022TOPDA2META
mmcrnsd: Processing disk DE00022TOPDA2DATA
mmcrnsd: Processing disk DE00022TOPDA3META
mmcrnsd: Processing disk DE00022TOPDA3DATA
mmcrnsd: Processing disk DE00022TOPDA4META
mmcrnsd: Processing disk DE00022TOPDA4DATA
mmcrnsd: Processing disk DE00022BOTDA1META
mmcrnsd: Processing disk DE00022BOTDA1DATA
mmcrnsd: Processing disk DE00022BOTDA2META
mmcrnsd: Processing disk DE00022BOTDA2DATA
mmcrnsd: Processing disk DE00022BOTDA3META
mmcrnsd: Processing disk DE00022BOTDA3DATA
mmcrnsd: Processing disk DE00022BOTDA4META
mmcrnsd: Processing disk DE00022BOTDA4DATA
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Notice how the recovery group log vdisks are omitted from NSD processing.

The **mmcrnsd** command then once again rewrites the stanza file in preparation for use as input to the **mmcrfs** command.

Creating the GPFS file system

Run the **mmcrfs** command to create the file system:

```
# mmcrfs gpfs -F mmcrvdisk.DE00022ALL -B 16m --metadata-block-size 1m -T /gpfs -A no

The following disks of gpfs will be formatted on node c250f09c01ap05.ppd.pok.ibm.com:
    DE00022TOPDA1META: size 262163456 KB
    DE00022TOPDA1DATA: size 8395522048 KB
    DE00022TOPDA2META: size 262163456 KB
    DE00022TOPDA2DATA: size 8395522048 KB
    DE00022TOPDA3META: size 262163456 KB
    DE00022TOPDA3DATA: size 8395522048 KB
    DE00022TOPDA4META: size 262163456 KB
    DE00022TOPDA4DATA: size 8395522048 KB
    DE00022BOTDA1META: size 262163456 KB
    DE00022BOTDA1DATA: size 8395522048 KB
    DE00022BOTDA2META: size 262163456 KB
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 215

```
    DE00022BOTDA2DATA: size 8395522048 KB
    DE00022BOTDA3META: size 262163456 KB
    DE00022BOTDA3DATA: size 8395522048 KB
    DE00022BOTDA4META: size 262163456 KB
    DE00022BOTDA4DATA: size 8395522048 KB
Formatting file system ...
Disks up to size 2.5 TB can be added to storage pool 'system'.
Disks up to size 79 TB can be added to storage pool 'data'.
Creating Inode File
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool 'system'
Formatting Allocation Map for storage pool 'data'
Completed creation of file system /dev/gpfs.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Notice how the 16 MiB data block size is specified with the traditional **-B** parameter and the 1 MiB metadata block size is specified with the **--metadata-block-size** parameter. Since a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool. In this example, the file placement policy is very simple:

```
# cat policy
rule 'default' set pool 'data'
```

The policy must then be installed in the file system using the **mmchpolicy** command:

```
# mmchpolicy gpfs policy -I yes
Validated policy `policy': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules
Policy `policy' installed and broadcast to all nodes.
```

If a policy is not placed in a file system with multiple storage pools, attempts to place data into files will return ENOSPC as if the file system were full.

This file system, built on a Power 775 Disk Enclosure using two recovery groups, two recovery group servers, eight file system metadata vdisk NSDs and eight file system data vdisk NSDs, may now be mounted and placed into service:

```
# mmmount gpfs -a
```

Example scenario: Replacing failed disks in a Power 775 Disk Enclosure recovery group

The scenario presented here shows how to detect and replace failed disks in a recovery group built on a Power 775 Disk Enclosure.

Detecting failed disks in your enclosure

Assume a fully populated Power 775 Disk Enclosure (serial number 000DE37) on which the following two recovery groups are defined:

- 000DE37TOP containing the disks in the top set of carriers
- 000DE37BOT containing the disks in the bottom set of carriers

Each recovery group contains the following:

- one log declustered array (LOG)
- four data declustered arrays (DA1, DA2, DA3, DA4)

The data declustered arrays are defined according to Power 775 Disk Enclosure best practice as follows:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 216

- 47 pdisks per data declustered array
- each member pdisk from the same carrier slot
- default disk replacement threshold value set to 2

The replacement threshold of 2 means that GPFS Native RAID will only require disk replacement when two or more disks have failed in the declustered array; otherwise, rebuilding onto spare space or reconstruction from redundancy will be used to supply affected data.

This configuration can be seen in the output of **mmlsrecoverygroup** for the recovery groups, shown here for `000DE37TOP`:

```
# mmlsrecoverygroup 000DE37TOP -L

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     192

 declustered   needs                              replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       63%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       19%  low
 DA3          yes           2      47       2          2       0   B   14 days  rebuild-2r  48%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       33%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       87%  low

                                    declustered
 vdisk               RAID code         array     vdisk size  remarks
 ------------------  ------------------  -----------  ----------  -------
 000DE37TOPLOG       3WayReplication     LOG            4144 MiB  log
 000DE37TOPDA1META   4WayReplication     DA1             250 GiB
 000DE37TOPDA1DATA   8+3p                DA1              17 TiB
 000DE37TOPDA2META   4WayReplication     DA2             250 GiB
 000DE37TOPDA2DATA   8+3p                DA2              17 TiB
 000DE37TOPDA3META   4WayReplication     DA3             250 GiB
 000DE37TOPDA3DATA   8+3p                DA3              17 TiB
 000DE37TOPDA4META   4WayReplication     DA4             250 GiB
 000DE37TOPDA4DATA   8+3p                DA4              17 TiB

 active recovery group server                      servers
 ------------------------------------------------  -------
 server1                                           server1,server2
```

The indication that disk replacement is called for in this recovery group is the value of `yes` in the `needs service` column for declustered array `DA3`.

The fact that `DA3` (the declustered array on the disks in carrier slot 3) is undergoing rebuild of its RAID tracks that can tolerate two strip failures is by itself not an indication that disk replacement is required; it merely indicates that data from a failed disk is being rebuilt onto spare space. Only if the replacement threshold has been met will disks be marked for replacement and the declustered array marked as needing service.

GPFS Native RAID provides several indications that disk replacement is required:

- entries in the AIX error report or the Linux syslog
- the GPFS **pdReplacePdisk** callback, which can be configured to run an administrator-supplied script at the moment a pdisk is marked for replacement
- the POWER7® cluster event notification TEAL agent, which can be configured to send disk replacement notices when they occur to the POWER7 cluster EMS

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 217

- the output from the following commands, which may be performed from the command line on any GPFS cluster node (see the examples that follow):
  1. **mmlsrecoverygroup** with the **-L** flag shows yes in the needs service column
  2. **mmlsrecoverygroup** with the **-L** and **--pdisk** flags; this shows the states of all pdisks, which may be examined for the replace pdisk state
  3. **mmlspdisk** with the **--replace** flag, which lists only those pdisks that are marked for replacement

Note: Because the output of **mmlsrecoverygroup -L --pdisk** for a fully-populated disk enclosure is very long, this example shows only some of the pdisks (but includes those marked for replacement).

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                  declustered
 recovery group      arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                  5       9     192

 declustered   needs                              replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       63%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       19%  low
 DA3          yes           2      47       2          2       0   B   14 days  rebuild-2r  68%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       34%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       87%  low

                   number of    declustered
 pdisk            active paths     array     free space  state
 -----------------  ------------  -----------  ----------  -----
 [...]
 c014d1                  2        DA1             23 GiB  ok
 c014d2                  2        DA2             23 GiB  ok
 c014d3                  2        DA3            510 GiB  dead/systemDrain/noRGD/noVCD/replace
 c014d4                  2        DA4             12 GiB  ok
 [...]
 c018d1                  2        DA1             24 GiB  ok
 c018d2                  2        DA2             24 GiB  ok
 c018d3                  2        DA3            558 GiB  dead/systemDrain/noRGD/noVCD/noData/replace
 c018d4                  2        DA4             12 GiB  ok
 [...]
```

The preceding output shows that the following pdisks are marked for replacement:

- c014d3 in DA3
- c018d3 and DA3

The naming convention used during recovery group creation indicates that these are the disks in slot 3 of carriers 14 and 18. To confirm the physical locations of the failed disks, use the **mmlspdisk** command to list information about those pdisks in declustered array DA3 of recovery group 000DE37TOP that are marked for replacement:

```
# mmlspdisk 000DE37TOP --declustered-array DA3 --replace
pdisk:
   replacementPriority = 1.00
   name = "c014d3"
   device = "/dev/rhdisk158,/dev/rhdisk62"
   recoveryGroup = "000DE37TOP"
   declusteredArray = "DA3"
   state = "dead/systemDrain/noRGD/noVCD/replace"
   freeSpace = 513745775616
   fru = "74Y4936"
   location = "78AD.001.000DE37-C14-D3"
   WWN = "naa.5000C5001D90E17F"
   server = "server1"
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 218

```
   reads = 986
   writes = 293368
   bytesReadInGiB = 0.826
   bytesWrittenInGiB = 286.696
   IOErrors = 0
   IOTimeouts = 12
   mediaErrors = 0
   checksumErrors = 0
   pathErrors = 0
   timeBadness = 33.278
   dataBadness = 0.000
pdisk:
   replacementPriority = 1.00
   name = "c018d3"
   device = "/dev/rhdisk630,/dev/rhdisk726"
   recoveryGroup = "000DE37TOP"
   declusteredArray = "DA3"
   state = "dead/systemDrain/noRGD/noVCD/noData/replace"
   freeSpace = 599147937792
   fru = "74Y4936"
   location = "78AD.001.000DE37-C18-D3"
   WWN = "naa.5000C5001DC5DF3F"
   server = "server1"
   reads = 1104
   writes = 379053
   bytesReadInGiB = 0.844
   bytesWrittenInGiB = 370.717
   IOErrors = 0
   IOTimeouts = 0
   mediaErrors = 0
   checksumErrors = 10
   pathErrors = 0
   timeBadness = 0.000
   dataBadness = 10.000
```

The preceding location code attributes confirm the pdisk naming convention:

Disk	Location code	Interpretation
pdisk `c014d3`	78AD.001.000DE37-C14-D3	Disk 3 in carrier 14 in the disk enclosure identified by enclosure type 78AD.001 and serial number 000DE37
pdisk `c018d3`	78AD.001.000DE37-C18-D3	Disk 3 in carrier 18 in the disk enclosure identified by enclosure type 78AD.001 and serial number 000DE37

Replacing the failed disks in a Power 775 Disk Enclosure recovery group

Note: In this example, it is assumed that two new disks with the appropriate Field Replaceable Unit (FRU) code, as indicated by the `fru` attribute (`74Y4936` in this case), have been obtained as replacements for the failed pdisks `c014d3` and `c018d3`.

Replacing each disk is a three-step process:

1. Using the **mmchcarrier** command with the **--release** flag to suspend use of the other disks in the carrier and to release the carrier.
2. Removing the carrier and replacing the failed disk within with a new one.
3. Using the **mmchcarrier** command with the **--replace** flag to resume use of the suspended disks and to begin use of the new disk.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 219

GPFS Native RAID assigns a priority to pdisk replacement. Disks with smaller values for the **replacementPriority** attribute should be replaced first. In this example, the only failed disks are in DA3 and both have the same **replacementPriority**.

Disk c014d3 is chosen to be replaced first.

1. To release carrier 14 in disk enclosure 000DE37:

```
# mmchcarrier 000DE37TOP --release --pdisk c014d3
  [I] Suspending pdisk c014d1 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D1.
  [I] Suspending pdisk c014d2 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D2.
  [I] Suspending pdisk c014d3 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D3.
  [I] Suspending pdisk c014d4 of RG 000DE37TOP in location 78AD.001.000DE37-C14-D4.
  [I] Carrier released.

    - Remove carrier.
    - Replace disk in location 78AD.001.000DE37-C14-D3 with FRU 74Y4936.
    - Reinsert carrier.
    - Issue the following command:

        mmchcarrier 000DE37TOP --replace --pdisk 'c014d3'

      Repair timer is running.  Perform the above within 5 minutes
      to avoid pdisks being reported as missing.
```

   GPFS Native RAID issues instructions as to the physical actions that must be taken. Note that disks may be suspended only so long before they are declared missing; therefore the mechanical process of physically performing disk replacement must be accomplished promptly.

   Use of the other three disks in carrier 14 has been suspended, and carrier 14 is unlocked. The identify lights for carrier 14 and for disk 3 are on.

2. Carrier 14 should be unlatched and removed. The failed disk 3, as indicated by the internal identify light, should be removed, and the new disk with FRU 74Y4936 should be inserted in its place. Carrier 14 should then be reinserted and the latch closed.

3. To finish the replacement of pdisk c014d3:

```
# mmchcarrier 000DE37TOP --replace --pdisk c014d3
[I] The following pdisks will be formatted on node server1:
    /dev/rhdisk354
[I] Pdisk c014d3 of RG 000DE37TOP successfully replaced.
[I] Resuming pdisk c014d1 of RG 000DE37TOP.
[I] Resuming pdisk c014d2 of RG 000DE37TOP.
[I] Resuming pdisk c014d3#162 of RG 000DE37TOP.
[I] Resuming pdisk c014d4 of RG 000DE37TOP.
[I] Carrier resumed.
```

When the **mmchcarrier --replace** command returns successfully, GPFS Native RAID has resumed use of the other 3 disks. The failed pdisk may remain in a temporary form (indicated here by the name c014d3#162) until all data from it has been rebuilt, at which point it is finally deleted. The new replacement disk, which has assumed the name c014d3, will have RAID tracks rebuilt and rebalanced onto it. Notice that only one block device name is mentioned as being formatted as a pdisk; the second path will be discovered in the background.

This can be confirmed with **mmlsrecoverygroup -L --pdisk**:

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     193

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       63%  low
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 220

```
DA2          no            2      47       2          2    3072 MiB   14 days  scrub       19%  low
DA3          yes           2      48       2          2       0   B   14 days  rebuild-2r  89%  low
DA4          no            2      47       2          2    3072 MiB   14 days  scrub       34%  low
LOG          no            1       4       1          1     546 GiB   14 days  scrub       87%  low

                  number of    declustered
pdisk             active paths    array     free space  state
----------------  ------------  -----------  ----------  -----
[...]
c014d1                 2        DA1             23 GiB  ok
c014d2                 2        DA2             23 GiB  ok
c014d3                 2        DA3            550 GiB  ok
c014d3#162             0        DA3            543 GiB  dead/adminDrain/noRGD/noVCD/noPath
c014d4                 2        DA4             23 GiB  ok
[...]
c018d1                 2        DA1             24 GiB  ok
c018d2                 2        DA2             24 GiB  ok
c018d3                 2        DA3            558 GiB  dead/systemDrain/noRGD/noVCD/noData/replace
c018d4                 2        DA4             23 GiB  ok
[...]
```

Notice that the temporary pdisk c014d3#162 is counted in the total number of pdisks in declustered array DA3 and in the recovery group, until it is finally drained and deleted.

Notice also that pdisk c018d3 is still marked for replacement, and that DA3 still needs service. This is because GPFS Native RAID replacement policy expects all failed disks in the declustered array to be replaced once the replacement threshold is reached. The replace state on a pdisk is not removed when the total number of failed disks goes under the threshold.

Pdisk c018d3 is replaced following the same process.

1. Release carrier 18 in disk enclosure 000DE37:

```
# mmchcarrier 000DE37TOP --release --pdisk c018d3
  [I] Suspending pdisk c018d1 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D1.
  [I] Suspending pdisk c018d2 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D2.
  [I] Suspending pdisk c018d3 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D3.
  [I] Suspending pdisk c018d4 of RG 000DE37TOP in location 78AD.001.000DE37-C18-D4.
  [I] Carrier released.

    - Remove carrier.
    - Replace disk in location 78AD.001.000DE37-C18-D3 with FRU 74Y4936.
    - Reinsert carrier.
    - Issue the following command:

        mmchcarrier 000DE37TOP --replace --pdisk 'c018d3'

      Repair timer is running.  Perform the above within 5 minutes
      to avoid pdisks being reported as missing.
```

2. Unlatch and remove carrier 18, remove and replace failed disk 3, reinsert carrier 18, and close the latch.
3. To finish the replacement of pdisk c018d3:

```
# mmchcarrier 000DE37TOP --replace --pdisk c018d3

  [I] The following pdisks will be formatted on node server1:
      /dev/rhdisk674
  [I] Pdisk c018d3 of RG 000DE37TOP successfully replaced.
  [I] Resuming pdisk c018d1 of RG 000DE37TOP.
  [I] Resuming pdisk c018d2 of RG 000DE37TOP.
  [I] Resuming pdisk c018d3#166 of RG 000DE37TOP.
  [I] Resuming pdisk c018d4 of RG 000DE37TOP.
  [I] Carrier resumed.
```

Running **mmlsrecoverygroup** again will confirm the second replacement:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 221

```
# mmlsrecoverygroup 000DE37TOP -L --pdisk

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 000DE37TOP                   5       9     192

 declustered   needs                              replace               scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 DA1          no            2      47       2          2    3072 MiB   14 days  scrub       64%  low
 DA2          no            2      47       2          2    3072 MiB   14 days  scrub       22%  low
 DA3          no            2      47       2          2    2048 MiB   14 days  rebalance   12%  low
 DA4          no            2      47       2          2    3072 MiB   14 days  scrub       36%  low
 LOG          no            1       4       1          1     546 GiB   14 days  scrub       89%  low

                     number of    declustered
 pdisk              active paths     array     free space  state
 -----------------  ------------  -----------  ----------  -----
 [...]
 c014d1                    2        DA1             23 GiB  ok
 c014d2                    2        DA2             23 GiB  ok
 c014d3                    2        DA3            271 GiB  ok
 c014d4                    2        DA4             23 GiB  ok
 [...]
 c018d1                    2        DA1             24 GiB  ok
 c018d2                    2        DA2             24 GiB  ok
 c018d3                    2        DA3            542 GiB  ok
 c018d4                    2        DA4             23 GiB  ok
 [...]
```

Notice that both temporary pdisks have been deleted. This is because `c014d3#162` has finished draining, and because pdisk `c018d3#166` had, before it was replaced, already been completely drained (as evidenced by the `noData` flag). Declustered array `DA3` no longer needs service and once again contains 47 pdisks, and the recovery group once again contains 192 pdisks.

GPFS Native RAID setup and maintenance on the IBM System x GPFS Storage Server (GSS)

The functions described in these topics are only available with the IBM System x GPFS Storage Server (GSS).

Updating firmware on enclosures and drives

After creating a GPFS cluster, you can configure the GSS.

The configuration process requires that you upgrade the enclosure and drive firmware.

Note: Before upgrading the enclosure and drive firmware, ensure that the `gpfs.gss.firmware` rpm has been installed on all GSS servers. `gpfs.gss.firmware` is a standard rpm that can be installed using the **rpm** command. It contains the latest updates of the following types of supported firmware for a GSS configuration:

- System BIOS updates
- Adapter firmware
- Enclosure firmware
- Driver firmware
- Firmware loading tools

To upgrade the enclosure firmware, follow these steps:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 222

1. To confirm that the enclosures have appropriate redundancy and that firmware updates have been found, issue this command:

   ```
   mmchfirmware --type storage-enclosure -N Node[,Node...] --dry-run
   ```

2. To load the firmware, issue this command:

   Note: If the existing firmware levels are less than 0375 (as shown by the **mmlsfirmware --type storage-enclosure** command issued in the previous step), include the **--stop-on-error no** option.

   ```
   mmchfirmware --type storage-enclosure -N Node[,Node...][--stop-on-error no]
   ```

To upgrade the drive firmware, follow these steps:

1. To confirm that the drives can be updated, issue this command:

   ```
   mmchfirmware --type drive -N Node[,Node...] --dry-run
   ```

2. To load the firmware, issue this command:

   ```
   mmchfirmware --type drive -N Node[,Node...]
   ```

Example scenario: Configuring GPFS Native RAID recovery groups on the GSS

This topic provides a detailed example of configuring GPFS Native RAID on a GSS building block. The example considers one GSS-24 building block, consisting of two recovery group servers cabled to four DCS3700 JBOD disk enclosures, and shows how recovery groups are defined on the disk enclosures. Throughout this topic, it may be helpful to have IBM System x GPFS Storage Server (GSS) documentation at hand.

Preparing GSS recovery group servers

Disk enclosure and HBA cabling

The GSS-24 DCS3700 JBOD disk enclosures should be cabled to the intended recovery group servers according to the IBM System x GPFS Storage Server (GSS) hardware installation instructions. The GSS-24 building block contains four disk enclosures. Each disk enclosure contains five disk drawers. In each disk enclosure, drawers 2, 3, and 4 contain 12 HDDs. In disk enclosures 1 through 3, drawers 1 and 5 have one SSD and 11 HDDs. In disk enclosure 4 (and, in the case of GSS-26, 5 and 6), drawers 1 and 5 have 11 HDDs and 1 empty slot. In total, a GSS-24 building block has 4 enclosures, 20 drawers, 232 HDDs, 6 SSDs, and 2 empty slots. (For comparison, a GSS-26 building block has 6 enclosures, 30 drawers, 348 HDDs, 6 SSDs, and 6 empty slots.) Each disk enclosure provides redundant A and B ESM port connections for host server HBA connections. To ensure proper multi-pathing and redundancy, each recovery group server must be connected to each of the A and B ESMs using different HBAs. The IBM System x GPFS Storage Server (GSS) hardware installation documentation describes precisely which HBA ports must be connected to which disk enclosure ESM ports.

GPFS Native RAID provides system administration tools for verifying the correct connectivity of GSS-24 and GSS-26 disk enclosures, which will be seen later during the operating system preparation.

When the ESM ports of the GSS-24 disk enclosures have been cabled to the appropriate HBAs of the two recovery group servers, the disk enclosures should be powered on and the recovery group servers should be rebooted.

Verifying that the GSS-24 building block is configured correctly

Preparation then continues at the operating system level on the recovery group servers, which must be installed with the Red Hat Enterprise Linux 6.3 distribution provided with the GSS. Once the cabling has been performed and the servers have been rebooted, it is necessary to perform a thorough discovery of the disk topology on each server.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 223

To proceed, GPFS must be installed on the recovery group servers, but they need not yet be added to a GPFS cluster. Consult *GPFS: Concepts, Planning, and Installation Guide* for instructions about installing GPFS.

GPFS Native RAID provides tools in `/usr/lpp/mmfs/samples/vdisk` for collecting and collating information on any attached GSS-24 disk enclosures and for verifying that the detected topology is correct. The **mmgetpdisktopology** command examines the list of connected devices for the operating system and produces a colon-delimited database with a line for each discovered GSS-24 physical disk, disk enclosure ESM expander device, and HBA. **mmgetpdisktopology** should be run on each of the two intended recovery group server nodes, and the results examined to verify that the disk enclosure hardware and software configuration is as expected. An additional tool called **topsummary** concisely summarizes the output of the **mmgetpdisktopology** command.

Create a directory in which to work, and then capture the output of the **mmgetpdisktopology** command from each of the two intended recovery group server nodes:

```
# mkdir gss-24
# cd gss-24
# ssh server1 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server1.top
# ssh server2 /usr/lpp/mmfs/samples/vdisk/mmgetpdisktopology > server2.top
```

Then view the summary for each of the nodes; the following example is for `server1`:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
DCS3700 enclosures found: SV21311179 SV21313978 SV21314035 SV21419522
Enclosure SV21314035 (number 1):
Enclosure SV21314035 ESM A sg243[0363][scsi8 port 4] ESM B sg62[0363][scsi7 port 4]
Enclosure SV21314035 Drawer 1 ESM sg243 12 disks diskset "11082" ESM sg62 12 disks diskset "11082"
Enclosure SV21314035 Drawer 2 ESM sg243 12 disks diskset "16380" ESM sg62 12 disks diskset "16380"
Enclosure SV21314035 Drawer 3 ESM sg243 12 disks diskset "11864" ESM sg62 12 disks diskset "11864"
Enclosure SV21314035 Drawer 4 ESM sg243 12 disks diskset "31717" ESM sg62 12 disks diskset "31717"
Enclosure SV21314035 Drawer 5 ESM sg243 12 disks diskset "11824" ESM sg62 12 disks diskset "11824"
Enclosure SV21314035 sees 60 disks
Enclosure SV21419522 (number 2):
Enclosure SV21419522 ESM A sg63[0363][scsi7 port 3] ESM B sg426[0363][scsi9 port 4]
Enclosure SV21419522 Drawer 1 ESM sg63 12 disks diskset "60508" ESM sg426 12 disks diskset "60508"
Enclosure SV21419522 Drawer 2 ESM sg63 12 disks diskset "29045" ESM sg426 12 disks diskset "29045"
Enclosure SV21419522 Drawer 3 ESM sg63 12 disks diskset "08276" ESM sg426 12 disks diskset "08276"
Enclosure SV21419522 Drawer 4 ESM sg63 12 disks diskset "28750" ESM sg426 12 disks diskset "28750"
Enclosure SV21419522 Drawer 5 ESM sg63 12 disks diskset "34265" ESM sg426 12 disks diskset "34265"
Enclosure SV21419522 sees 60 disks
Enclosure SV21313978 (number 3):
Enclosure SV21313978 ESM A sg365[0363][scsi9 port 3] ESM B sg244[0363][scsi8 port 3]
Enclosure SV21313978 Drawer 1 ESM sg365 12 disks diskset "36679" ESM sg244 12 disks diskset "36679"
Enclosure SV21313978 Drawer 2 ESM sg365 12 disks diskset "18808" ESM sg244 12 disks diskset "18808"
Enclosure SV21313978 Drawer 3 ESM sg365 12 disks diskset "55525" ESM sg244 12 disks diskset "55525"
Enclosure SV21313978 Drawer 4 ESM sg365 12 disks diskset "51485" ESM sg244 12 disks diskset "51485"
Enclosure SV21313978 Drawer 5 ESM sg365 12 disks diskset "24274" ESM sg244 12 disks diskset "24274"
Enclosure SV21313978 sees 60 disks
Enclosure SV21311179 (number 4):
Enclosure SV21311179 ESM A sg184[0363][scsi8 port 2] ESM B sg3[0363][scsi7 port 2]
Enclosure SV21311179 Drawer 1 ESM sg184 11 disks diskset "59007" ESM sg3 11 disks diskset "59007"
Enclosure SV21311179 Drawer 2 ESM sg184 12 disks diskset "01848" ESM sg3 12 disks diskset "01848"
Enclosure SV21311179 Drawer 3 ESM sg184 12 disks diskset "49359" ESM sg3 12 disks diskset "49359"
Enclosure SV21311179 Drawer 4 ESM sg184 12 disks diskset "62742" ESM sg3 12 disks diskset "62742"
Enclosure SV21311179 Drawer 5 ESM sg184 11 disks diskset "62485" ESM sg3 11 disks diskset "62485"
Enclosure SV21311179 sees 58 disks
DCS3700 configuration: 4 enclosures, 6 SSDs, 2 empty slots, 238 disks total
scsi7[14.00.01.00] 0000:11:00.0 [P2 SV21311179 ESM B (sg3)] [P3 SV21419522 ESM A (sg63)] [P4 SV21314035 ESM B (sg62)]
scsi8[14.00.01.00] 0000:8b:00.0 [P2 SV21311179 ESM A (sg184)] [P3 SV21313978 ESM B (sg244)] [P4 SV21314035 ESM A (sg243)]
scsi9[14.00.01.00] 0000:90:00.0 [P3 SV21313978 ESM A (sg365)] [P4 SV21419522 ESM B (sg426)]
```

This output shows a correctly cabled and populated GSS-24 installation. Four disk enclosures with serial numbers `SV21311179`, `SV21313978`, `SV21314035`, and `SV21419522` are found cabled in the correct order and with the correct numbers of disks.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 224

The section for each enclosure begins with the enclosure number as determined by the cabling, followed by which ESM and HBA ports provide connectivity to the enclosure; the following line indicates that /dev/sg184 is the SCSI/SES expander device representing the enclosure's A ESM, which is at firmware level 0363 and is connected to port 2 of the HBA that the operating system configured as SCSI host 8:

```
Enclosure SV21311179 ESM A sg184[0363][scsi8 port 2] ESM B sg3[0363][scsi7 port 2]
```

Each drawer of an enclosure is represented by a line indicating how many disks are seen over each ESM and a "diskset" checksum of the disk WWNs, which is used to verify that each ESM connection sees the same set of disks over different device paths.

If the cabling were incorrect in any way, one or more of the enclosures would indicate "number undetermined," and the following message would be printed:

```
Unable to determine enclosure order; check the HBA to enclosure cabling.
```

If any slots are unexpectedly empty or contain the wrong type of disk, the **topsummary** output will include additional messages such as these:

```
Location SV21311179-2-10 appears only on the sg184 path
Location SV21313978-1-3 appears empty but should have an SSD
Location SV21311179-5-12 has an HDD but should be empty
```

The HBAs are also listed (firmware levels in brackets) with their bus addresses and port connections. For example, [P2 SV21311179 ESM B (sg3)] indicates that HBA port 2 is connected to ESM B of enclosure SV21311179; ESM B is represented by the /dev/sg3 SCSI/SES expander device. The HBA bus addresses and port connections are standard in a GSS building block and are used by **topsummary** to validate the correctness of the cabling.

If the cabling is called out as incorrect and the enclosure order cannot be determined, or if other discrepancies are noted (for example, physical disks that are expected but do not show up at all, or SSDs or HDDs in the wrong locations), corrections should be made and verified before proceeding. This would probably involve shutting down the servers and the disk enclosures and fixing the cabling and disk placement.

The server2.top topology database should also be examined with the **topsummary** sample script and verified to be correct. It should also be verified to have found the same enclosures with the same serial numbers and in the same order. One way to do this is to run the following commands and verify that they give exactly the same output:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top | grep number
# /usr/lpp/mmfs/samples/vdisk/topsummary server2.top | grep number
```

Once the GSS-24 disk enclosure topologies are verified to be correct on both intended recovery group server nodes, it is highly recommended that the disk enclosures be entered into the GPFS Native RAID component database. This allows the system administrator to provide a meaningful name for each component and to record each of their physical rack locations in a manner intended to make maintenance actions easier.

To proceed (either to populate the GPFS Native RAID component database or to create recovery groups), the recovery group servers must now be members of the same GPFS cluster. Consult *GPFS: Administration and Programming Reference* for instructions for creating a GPFS cluster.

Once the intended recovery group servers are added to a GPFS cluster, the component database can be updated to identify the equipment rack and the U compartments in which the disk enclosures reside.

In this scenario, a GSS-24 building block will occupy an Intelligent Cluster 42U Primary Rack. Using the component database, the four disk enclosures and the rack in which they reside may be given meaningful names and associated with each other.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 225

To create the component database entry for an Intelligent Cluster 42U Primary Rack, which has part number 1410HPA, and to give it the descriptive name BB1, use the **mmaddcomp** command:

```
# mmaddcomp 1410HPA --name BB1
```

Do the same for each of the four disk enclosures as identified by the **topsummary** command. In this example, the enclosures were identified according to the cabling in this order:

```
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top | grep number
Enclosure SV21314035 (number 1):
Enclosure SV21419522 (number 2):
Enclosure SV21313978 (number 3):
Enclosure SV21311179 (number 4):
```

Suppose this is GSS-24 building block 1, and the desired descriptive names for the disk enclosures are BB1ENC1, BB1ENC2, BB1ENC3, and BB1ENC4. To define these to the component database, use the disk enclosure part number 181880E as follows:

```
# mmaddcomp 181880E --serial-number SV21314035 --name BB1ENC1
# mmaddcomp 181880E --serial-number SV21419522 --name BB1ENC2
# mmaddcomp 181880E --serial-number SV21313978 --name BB1ENC3
# mmaddcomp 181880E --serial-number SV21311179 --name BB1ENC4
```

You may also use the **mmdiscovercomp** command in place of **mmaddcomp**, then use **mmchcomp** to assign descriptive names to the enclosures.

At this point, the building block rack and disk enclosures can be listed from component database using the **mmlscomp** command:

```
# mmlscomp
Rack Components
Comp ID Part Number Serial Number Name
------- ----------- ------------- ------
      1 1410HPA                   BB1

Storage Enclosure Components

Comp ID Part Number Serial Number Name    Display ID
------- ----------- ------------- ------- ----------
      2 181880E     SV21314035    BB1ENC1
      3 181880E     SV21419522    BB1ENC2
      4 181880E     SV21313978    BB1ENC3
      5 181880E     SV21311179    BB1ENC4
```

Each of the defined components has an ID. The location of the enclosures can be defined according to the four U compartments they occupy in the rack. To do this, use the **mmchcomploc** command to specify the component IDs of the enclosures, the rack that contains them, and the starting U compartment (position) within the rack. The syntax of the **mmchcomploc** command is:

mmchcomploc *Component*
Container Position

To define enclosure BB1ENC1 as occupying U compartments 1 through 4 in rack BB1, use this command:

```
# mmchcomploc BB1ENC1 BB1 1
```

Suppose BB1ENC2, BB1ENC3, and BB1ENC4 have starting U compartments 5, 13, and 17, respectively, in rack BB1. Use the following commands to define these relationships:

```
# mmchcomploc BB1ENC2 BB1 5
# mmchcomploc BB1ENC3 BB1 13
# mmchcomploc BB1ENC4 BB1 17
```

The defined component locations can be listed with the **mmlscomploc** command:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 226

```
# mmlscomploc
Component Location
--------- ------------------
BB1ENC1   Rack BB1 U01-04
BB1ENC2   Rack BB1 U05-08
BB1ENC3   Rack BB1 U13-16
BB1ENC4   Rack BB1 U17-20
```

(U compartments 9 - 10 and 11 - 12 are presumably occupied by the recovery group servers. They are omitted here as extraneous to this set of examples. They may be named and defined if desired by using component part number 7915AC1 with the **mmaddcomp** command, and then using their component IDs and starting U compartments with the **mmchcomploc** command, as was done with the disk enclosures.)

With the above component names and locations having been entered into the GPFS Native RAID component database, a common maintenance action such as replacing a disk will be able to supply meaningful direction to the user in locating the equipment in a crowded machine room.

Creating recovery groups on the IBM System x GPFS Storage Server (GSS)

Configuring GPFS nodes to be recovery group servers

Having verified the disk enclosure connectivity of the GSS-24 building block, and having optionally also created a component database to associate names and machine room locations with the storage hardware, GPFS Native RAID must be enabled on the intended recovery group servers.

To proceed, the recovery group servers must be members of the same GPFS cluster. If this has not already been done in preparing a GPFS Native RAID component database, consult *GPFS: Administration and Programming Reference* for instructions for adding nodes to or creating a GPFS cluster.

Before a GPFS node can create and serve recovery groups, it must be configured with a vdisk track cache. This is accomplished by setting the **nsdRAIDTracks** configuration parameter.

nsdRAIDTracks is the GPFS configuration parameter essential to define a GPFS cluster node as a recovery group server. It specifies the number of vdisk tracks of which the attributes will be held in memory by the GPFS daemon on the recovery group server.

The actual contents of the vdisk tracks, the user data and the checksums, are stored during runtime in the standard GPFS pagepool. Therefore, the size of the GPFS pagepool configured on a recovery group server should be considerable, on the order of tens of gigabytes. The amount of pagepool dedicated to hold vdisk track data is governed by the **nsdRAIDBufferPoolSizePct** parameter, which defaults to 50%. In practice, a recovery group server will not need to use the GPFS pagepool for any significant amount of standard file caching, and the **nsdRAIDBufferPoolSizePct** value can be increased to 80%. This implies that the **prefetchPct** pagepool parameter can be reduced to as little as 5%. Since a recovery group server is also an NSD server, the vdisk buffer pool doubles as the NSD buffer pool. To achieve good NSD server performance, a large number of dedicated NSD service threads should be configured by setting **nsdMinWorkerThreads** and **nsdMaxWorkerThreads** to the same value, on the order of a few thousand.

In this example, the recovery group servers will be configured to cache the information on 131072 vdisk tracks and to have 38 GiB of pagepool, of which 80% will be used for vdisk data and 5% for application file caching. The servers are also configured to have 3072 NSD service threads. Once the configuration changes are made, the servers will need to be restarted.

```
# mmchconfig nsdRAIDTracks=131072,nsdRAIDBufferPoolSizePct=80,prefetchPct=5,pagepool=38G -N server1,server2
# mmchconfig nsdMinWorkerThreads=3072,nsdMaxWorkerThreads=3072 -N server1,server2
# mmshutdown -N server1,server2
# mmstartup -N server1,server2
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 227

Defining the recovery group layout

The definition of recovery groups on a GSS-24 building block is accomplished by dividing the drawers of the enclosures into left and right halves. The sharing of GSS-24 disk enclosures by two servers implies two recovery groups; one is served by one node and one by the other, and each server acts as the other's backup. Half the disks in each enclosure and drawer should belong to one recovery group, and half to the other. One recovery group will therefore be defined on the disks in the left half of each drawer, slots 1 though 6, and one on the disks in the right half of each drawer, slots 7 through 12. The three SSDs in drawer 1 slot 3 of the first three enclosures will make up the vdisk log declustered array for the left recovery group, and the three SSDs in drawer 5 slot 12 of the first three enclosures will make up the vdisk log declustered array of the right recovery group. The remaining 116 HDDs are divided into two vdisk data declustered arrays of 58 disks.

GPFS Native RAID provides a tool that understands the layout of GSS building blocks and will automatically generate the **mmcrrecoverygroup** stanza files for creating the left and right recovery groups. `/usr/lpp/mmfs/samples/vdisk/mkrginput`, when supplied with output of the **mmgetpdisktopology** command, will create recovery group stanza files for the left and right sets of disks as described in the previous paragraph.

Each recovery group server, though it may see the same functional disk enclosure topology, will almost certainly differ in the particulars of which disk device names (for example, `/dev/sdax`) refer to which physical disks in what disk enclosure location.

There are therefore two possibilities for creating the recovery group stanza files and the recovery groups themselves:

Alternative 1:
Generate the recovery group stanza files and create the recovery groups from the perspective of just one of the servers as if that server were to be primary for both recovery groups, and then use the **mmchrecoverygroup** command to swap the primary and backup servers for one of the recovery groups

Alternative 2:
Generate the recovery group stanza files for each server's primary recovery group using the primary server's topology file.

The following example shows both alternatives.

Creating the recovery groups, alternative 1
To create the recovery group input stanza files from the perspective of `server1`, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server1.top
```

This will create two files, one for the left set of disks and one for the right set of disks found in the `server1` topology. The files will be named after the serial number of the enclosure determined to be first in the topology, but each will contain disks from all four enclosures. In this case, the resulting stanza files will be `SV21314035L.server1` for the left half and `SV21314035R.server1` for the right half.

The recovery group stanza files will follow the recommended best practice for a GSS-24 building block of defining in each half a separate declustered array of three SSDs for recovery group transaction logging, and two file system data declustered arrays using the regular HDDs. One of the data declustered arrays will contain the disks from the left or right halves of enclosures 1 and 2, and the other will contain the disks from the left or right halves of enclosures 3 and 4. The vdisk log declustered array will be named `LOG`, and data declustered arrays will be named `DA1` and `DA2`.

Except for the `LOG` declustered array, the defaults are accepted for recovery group declustered array parameters such as scrub duration, spare space, and disk replacement policy. For the `LOG`

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 228

declustered array, the spare space will be defined to be 0. This will permit the log vdisk to later be created with three-way replication, which is a two-fault tolerant RAID code.

The stanza file will look something like this:

```
# head -14 SV21314035L.server1
%da: daName=LOG
     spares=0
%pdisk: pdiskName=e1d1s03log
        device=/dev/sdbk
        da=LOG
%pdisk: pdiskName=e2d1s03log
        device=/dev/sdbw
        da=LOG
%pdisk: pdiskName=e3d1s03log
        device=/dev/sdis
        da=LOG
%pdisk: pdiskName=e1d1s01
        device=/dev/sdbi
        da=DA1
```

All the pdisk stanzas for declustered array LOG will be listed first, followed by those for DA1 and DA2. The pdisk names will indicate the enclosure number, the drawer number, and the slot within the drawer in which the physical disk resides. The names of the pdisks in the LOG declustered array will have "log" appended to make them stand out. Notice that only one block device path to the disk is given; the second path will be discovered automatically by GPFS Native RAID soon after the recovery group is created.

If, as was recommended, the GPFS Native RAID component database was used to provide meaningful names for the GSS-24 building block components, the names of the recovery groups should be chosen to follow that convention. In this example, the building block rack was named BB1 and the enclosures were named BB1ENC1 through BB1ENC4. It would make sense then to name the left and right recovery groups BB1RGL and BB1RGR. Other conventions are of course possible.

With the left and right recovery group stanza files both created from the disk device perspective of server node server1, these two recovery groups can be created using two separate invocations of the **mmcrrecoverygroup** command:

```
# mmcrrecoverygroup BB1RGL -F SV21314035L.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
# mmcrrecoverygroup BB1RGR -F SV21314035R.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

Note that both recovery groups were created with server1 as primary and server2 as backup. It is now necessary to swap the primary and backup servers for BB1RGR using the **mmchrecoverygroup** command:

```
# mmchrecoverygroup BB1RGR --servers server2,server1
mmchrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

GPFS Native RAID will automatically discover the appropriate disk devices on server2.

Creating the recovery groups, alternative 2

To create the recovery groups from the start with the intended primary and backup servers, the stanza files from both server topologies will need to be created.

To create the server1 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server1.top
```

To create the server2 recovery group input stanza files, run:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server2.top
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 229

These two commands will result in four stanza files: SV21314035L.server1, SV21314035R.server1, SV21314035L.server2, and SV21314035R.server2.

The BB1RGL recovery group must then be created using server1 as the primary and the SV21314035L.server1 stanza file. The BB1RGR recovery group must be created using server2 as the primary and the SV21314035R.server2 stanza file.

```
# mmcrrecoverygroup BB1RGL -F SV21314035L.server1 --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
# mmcrrecoverygroup BB1RGR -F SV21314035R.server2 --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

Since each recovery group was created using the intended primary server and the stanza file for that server, it is not necessary to swap the primary and backup servers.

Verifying recovery group creation

Use the **mmlsrecoverygroup** command to verify that each recovery group was created:

```
# mmlsrecoverygroup BB1RGL -L

                 declustered
recovery group    arrays      vdisks pdisks
----------------- ----------- ------ ------
BB1RGL                      3      0    119
declustered needs                         replace              scrub      background activity
array       service vdisks pdisks spares threshold free space duration task   progress  priority
----------- ------- ------ ------ ------ --------- ---------- -------- -------------------------
LOG         no            0      3      0         1    558 GiB  14 days inactive     0%  low
DA1         no            0     58      2         2    101 TiB  14 days inactive     0%  low
DA2         no            0     58      2         2    101 TiB  14 days inactive     0%  low

                                      declustered                       checksum
vdisk               RAID code            array    vdisk size block size granularity remarks
------------------ ------------------ ----------- ---------- ---------- ----------- -------

active recovery group server                    servers
----------------------------------------------- -------
server1                                         server1,server2

# mmlsrecoverygroup BB1RGR -L
                 declustered
recovery group      arrays    vdisks pdisks
----------------- ----------- ------ ------
BB1RGR                      3      0    119
declustered needs                         replace              scrub    background activity
array       service vdisks pdisks spares threshold free space duration task   progress  priority
----------- ------- ------ ------ ------ --------- ---------- -------- -------------------------
LOG         no            0      3      0         1    558 GiB  14 days inactive     0%  low
DA1         no            0     58      2         2    101 TiB  14 days inactive     0%  low
DA2         no            0     58      2         2    101 TiB  14 days inactive     0%  low

                                      declustered                       checksum
vdisk               RAID code            array    vdisk size block size granularity remarks
------------------ ------------------ ----------- ---------- ---------- ----------- -------

active recovery group server                    servers
----------------------------------------------- -------
server1                                        server2,server1
```

Notice that the vdisk sections of the newly created recovery groups are empty; the next step is to create the vdisks.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 230

Defining and creating the vdisks

Once the recovery groups are created and being served by their respective servers, it is time to create the vdisks using the **mmcrvdisk** command.

Each recovery group requires a single log vdisk for recording RAID updates and diagnostic information. This is internal to the recovery group, cannot be used for user data, and should be the only vdisk in the `LOG` declustered array. The log vdisks use three-way replication in order to be fully two-fault tolerant in the `LOG` declustered array, which contains three SSDs and no spare space.

Data vdisks are required to be defined in the two data declustered arrays for use as file system NSDs. In this example, each of the declustered arrays for file system data is divided into two vdisks with different characteristics:

- one using 4-way replication and a 1 MiB block size and a total vdisk size of 1024 GiB suitable for file system metadata
- one using Reed-Solomon 8 + 3p encoding and an 16 MiB block size suitable for file system data

The vdisk size is omitted for the Reed-Solomon vdisks, meaning that they will default to use the remaining non-spare space in the declustered array (for this to work, any vdisks with specified total sizes must of course be defined first).

The possibilities for the vdisk creation stanza file are quite great, depending on the number and type of vdisk NSDs required for the number and type of file systems desired, so the vdisk stanza file will need to be created by hand, possibly following a template. The example vdisk stanza file that is supplied in `/usr/lpp/mmfs/samples/vdisk/vdisk`.stanza may be used for this purpose and adapted to specific file system requirements.

For GSS, it is suggested that each recovery group use the option of defining a log tip vdisk together with the mandatory log home vdisk. These vdisks record RAID updates and diagnostic information, are internal to each recovery group, and cannot be used for user data. The log tip vdisk is a fast staging area for RAID updates. When a log tip vdisk is used, it must be the only vdisk in the LOG declustered array, and must be the first vdisk defined in each recovery group. When log tip is used, the log home vdisk must be the second vdisk defined in each recovery group, and should be placed in the first data declustered array (in this example, `DA1`). Any additional data declustered arrays may not contain log vdisks. As a staging area, the log tip vdisk is relatively small in size at 128 MiB.

In this example, a single stanza file, `mmcrvdisk.BB1`, is used. The single file contains the specifications for all the vdisks in both the BB1RGL and BB1RGR recovery groups. Here is what the example stanza file for use with **mmcrvdisk** should look like:

```
# cat mmcrvdisk.BB1
%vdisk: vdiskName=BB1RGLLOGTIP
        rg=BB1RGL
        da=LOG
        blocksize=1m
        size=128m
        raidCode=3WayReplication
        diskUsage=vdiskLogTip
%vdisk: vdiskName=BB1RGRLOGTIP
        rg=BB1RGR
        da=LOG
        blocksize=1m
        size=128m
        raidCode=3WayReplication
        diskUsage=vdiskLogTip
%vdisk: vdiskName=BB1RGLLOGHOME
        rg=BB1RGL
        da=DA1
        blocksize=1m
        size=40g
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 231

```
        raidCode=4WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=BB1RGRLOGHOME
        rg=BB1RGR
        da=DA1
        blocksize=1m
        size=40g
        raidCode=4WayReplication
        diskUsage=vdiskLog
%vdisk: vdiskName=BB1RGLMETA1
        rg=BB1RGL
        da=DA1
        blocksize=1m
        size=256g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=1
        pool=system
%vdisk: vdiskName=BB1RGRMETA1
        rg=BB1RGR
        da=DA1
        blocksize=1m
        size=256g
        raidCode=4WayReplication
        diskUsage=metadataOnly
        failureGroup=1
        pool=system
%vdisk: vdiskName=BB1RGLDATA1
        rg=BB1RGL
        da=DA1
        blocksize=16m
        size=8192g
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=1
        pool=data
%vdisk: vdiskName=BB1RGRDATA1
        rg=BB1RGR
        da=DA1
        blocksize=16m
        size=8192g
        raidCode=8+3p
        diskUsage=dataOnly
        failureGroup=1
        pool=data
[DA2 vdisks omitted.]
```

Notice how the file system metadata vdisks are flagged for eventual file system usage as **metadataOnly** and for placement in the system storage pool, and the file system data vdisks are flagged for eventual **dataOnly** usage in the data storage pool. (After the file system is created, a policy will be required to allocate file system data to the correct storage pools; see “Creating the GPFS file system” on page 211.)

Importantly, also notice that block sizes for the file system metadata and file system data vdisks must be specified at this time, may not later be changed, and must match the block sizes supplied to the eventual **mmcrfs** command.

Notice also that the eventual **failureGroup=1** value for the NSDs on the file system vdisks is the same for vdisks in both the BB1RGL and BB1RGR recovery groups. This is because the recovery groups, although they have different servers, still share a common point of failure in the four GSS-24 disk enclosures, and GPFS should be informed of this through a distinct failure group designation for each disk enclosure. It is up to the GPFS system administrator to decide upon the failure group numbers for each GSS building block in the GPFS cluster. In this example, the failure group number 1 has been chosen to match the example building block number.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 232

To create the vdisks specified in the mmcrvdisk.BB1 file, use the following **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisk.BB1
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGRLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGLMETA1
mmcrvdisk: [I] Processing vdisk BB1RGRMETA1
mmcrvdisk: [I] Processing vdisk BB1RGLDATA1
mmcrvdisk: [I] Processing vdisk BB1RGRDATA1
mmcrvdisk: [I] Processing vdisk BB1RGLMETA2
mmcrvdisk: [I] Processing vdisk BB1RGRMETA2
mmcrvdisk: [I] Processing vdisk BB1RGLDATA2
mmcrvdisk: [I] Processing vdisk BB1RGRDATA2
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Creation of the vdisks may be verified through the **mmlsvdisk** command (the **mmlsrecoverygroup** command may also be used):

```
# mmlsvdisk
                                                    declustered  block size
 vdisk name          RAID code        recovery group       array       in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 BB1RGLDATA1         8+3p             BB1RGL              DA1               16384
 BB1RGLDATA2         8+3p             BB1RGL              DA2               16384
 BB1RGLLOGHOME       4WayReplication  BB1RGL              DA1                1024  log
 BB1RGLLOGTIP        3WayReplication  BB1RGL              LOG                1024  logTip
 BB1RGLMETA1         4WayReplication  BB1RGL              DA1                1024
 BB1RGLMETA2         4WayReplication  BB1RGL              DA2                1024
 BB1RGRDATA1         8+3p             BB1RGR              DA1               16384
 BB1RGRDATA2         8+3p             BB1RGR              DA2               16384
 BB1RGRLOGHOME       4WayReplication  BB1RGR              DA1                1024  log
 BB1RGRLOGTIP        3WayReplication  BB1RGR              LOG                1024  logTip
 BB1RGRMETA1         4WayReplication  BB1RGR              DA1                1024
 BB1RGRMETA2         4WayReplication  BB1RGR              DA2                1024
```

Creating NSDs from vdisks

The **mmcrvdisk** command rewrites the input file so that it is ready to be passed to the **mmcrnsd** command that creates the NSDs from which GPFS builds file systems. To create the vdisk NSDs, run the **mmcrnsd** command on the rewritten **mmcrvdisk** stanza file:

```
# mmcrnsd -F mmcrvdisk.BB1
mmcrnsd: Processing disk BB1RGLMETA1
mmcrnsd: Processing disk BB1RGRMETA1
mmcrnsd: Processing disk BB1RGLDATA1
mmcrnsd: Processing disk BB1RGRDATA1
mmcrnsd: Processing disk BB1RGLMETA2
mmcrnsd: Processing disk BB1RGRMETA2
mmcrnsd: Processing disk BB1RGLDATA2
mmcrnsd: Processing disk BB1RGRDATA2
mmcrnsd: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

Notice how the recovery group log vdisks are omitted from NSD processing.

The **mmcrnsd** command then once again rewrites the stanza file in preparation for use as input to the **mmcrfs** command.

Creating the GPFS file system

Run the **mmcrfs** command to create the file system:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 233

```
# mmcrfs gpfsbb1 -F mmcrvdisk.BB1 -B 16m --metadata-block-size 1m -T /gpfsbb1 -n 256
The following disks of gpfsbb1 will be formatted on node server1:
    BB1RGLMETA1: size 269213696 KB
    BB1RGRMETA1: size 269213696 KB
    BB1RGLDATA1: size 8593965056 KB
    BB1RGRDATA1: size 8593965056 KB
    BB1RGLMETA2: size 269213696 KB
    BB1RGRMETA2: size 269213696 KB
    BB1RGLDATA2: size 8593965056 KB
    BB1RGRDATA2: size 8593965056 KB
Formatting file system ...
Disks up to size 3.3 TB can be added to storage pool system.
Disks up to size 82 TB can be added to storage pool data.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
98 % complete on Sun Nov 18 13:27:00 2012
100 % complete on Sun Nov 18 13:27:00 2012
Formatting Allocation Map for storage pool data
85 % complete on Sun Nov 18 13:27:06 2012
100 % complete on Sun Nov 18 13:27:06 2012
Completed creation of file system /dev/gpfsbb1.
mmcrfs: Propagating the cluster configuration data to all
affected nodes. This is an asynchronous process.
```

(The **-n 256** parameter specifies that the allocation maps should account for 256 nodes mounting the file system. This is an example only and should be adjusted to actual cluster expectations.)

Notice how the 16 MiB data block size is specified with the traditional **-B** parameter and the 1 MiB metadata block size is specified with the **--metadata-block-size** parameter. Since a file system with different metadata and data block sizes requires the use of multiple GPFS storage pools, a file system placement policy is needed to direct user file data to the data storage pool. In this example, the file placement policy is very simple:

```
# cat policy
rule 'default' set pool 'data'
```

The policy must then be installed in the file system using the **mmchpolicy** command:

```
# mmchpolicy gpfsbb1 policy -I yes
Validated policy `policy': parsed 1 Placement Rules, 0 Restore Rules, 0 Migrate/Delete/Exclude Rules,
        0 List Rules, 0 External Pool/List Rules
Policy `policy' installed and broadcast to all nodes.
```

If a policy is not placed in a file system with multiple storage pools, attempts to place data into files will return ENOSPC as if the file system were full.

This file system, built on a GSS-24 building block using two recovery groups, two recovery group servers, four file system metadata vdisk NSDs and four file system data vdisk NSDs, may now be mounted and placed into service:

```
# mmmount gpfsbb1 -a
```

Differences in GSS 1.5 with LogTip NVRAM partitions

Disk enclosure and HBA cabling

GSS 1.5 servers are configured with six small (2000 MiB) disk partitions on the mirrored boot disk. These disk partitions are write-back storage for very fast NVRAM cache in the server HBAs. On each server in

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 234

a GSS building block, two of these NVRAM write-back partitions are used for the log tip declustered array for GNR RAID updates. (Event logs are only written to the log home vdisk on spinning disks, not the log tip vdisk.)

The **mmgetpdisktopology** command acquires information about NVRAM partitions on the server on which it is run. The **topsummary** command reports the existence of NVRAM partitions in the **mmgetpdisktopology** output.

For example, in the following output, the line "GSS configuration" indicates that there are six NVRAM partitions in this server topology. This is the expected output for a GSS 1.5 server in a GSS building block.

```
# ssh server1 /usr/lpp/mmfs/bin/mmgetpdisktopology > server1.top
# /usr/lpp/mmfs/samples/vdisk/topsummary server1.top
GSS enclosures found: SV21311179 SV21313978 SV21314035 SV21419522
Enclosure SV21314035 (number 1):
Enclosure SV21314035 ESM A sg248[0393][scsi8 port 4] ESM B sg67[0393][scsi7 port 4]
Enclosure SV21314035 Drawer 1 ESM sg248 12 disks diskset "13336" ESM sg67 12 disks diskset "13336"
Enclosure SV21314035 Drawer 2 ESM sg248 12 disks diskset "30460" ESM sg67 12 disks diskset "30460"
Enclosure SV21314035 Drawer 3 ESM sg248 12 disks diskset "55034" ESM sg67 12 disks diskset "55034"
Enclosure SV21314035 Drawer 4 ESM sg248 12 disks diskset "26111" ESM sg67 12 disks diskset "26111"
Enclosure SV21314035 Drawer 5 ESM sg248 12 disks diskset "34371" ESM sg67 12 disks diskset "34371"
Enclosure SV21314035 sees 60 disks

Enclosure SV21419522 (number 2):
Enclosure SV21419522 ESM A sg68[0393][scsi7 port 3] ESM B sg431[0393][scsi9 port 4]
Enclosure SV21419522 Drawer 1 ESM sg68 12 disks diskset "05350" ESM sg431 12 disks diskset "05350"
Enclosure SV21419522 Drawer 2 ESM sg68 12 disks diskset "29524" ESM sg431 12 disks diskset "29524"
Enclosure SV21419522 Drawer 3 ESM sg68 12 disks diskset "12828" ESM sg431 12 disks diskset "12828"
Enclosure SV21419522 Drawer 4 ESM sg68 12 disks diskset "52289" ESM sg431 12 disks diskset "52289"
Enclosure SV21419522 Drawer 5 ESM sg68 12 disks diskset "57634" ESM sg431 12 disks diskset "57634"
Enclosure SV21419522 sees 60 disks

Enclosure SV21313978 (number 3):
Enclosure SV21313978 ESM A sg370[0393][scsi9 port 3] ESM B sg249[0393][scsi8 port 3]
Enclosure SV21313978 Drawer 1 ESM sg370 12 disks diskset "09002" ESM sg249 12 disks diskset "09002"
Enclosure SV21313978 Drawer 2 ESM sg370 12 disks diskset "27263" ESM sg249 12 disks diskset "27263"
Enclosure SV21313978 Drawer 3 ESM sg370 12 disks diskset "14213" ESM sg249 12 disks diskset "14213"
Enclosure SV21313978 Drawer 4 ESM sg370 12 disks diskset "36478" ESM sg249 12 disks diskset "36478"
Enclosure SV21313978 Drawer 5 ESM sg370 12 disks diskset "31136" ESM sg249 12 disks diskset "31136"
Enclosure SV21313978 sees 60 disks

Enclosure SV21311179 (number 4):
Enclosure SV21311179 ESM A sg189[0393][scsi8 port 2] ESM B sg8[0393][scsi7 port 2]
Enclosure SV21311179 Drawer 1 ESM sg189 11 disks diskset "18272" ESM sg8 11 disks diskset "18272"
Enclosure SV21311179 Drawer 2 ESM sg189 12 disks diskset "61983" ESM sg8 12 disks diskset "61983"
Enclosure SV21311179 Drawer 3 ESM sg189 12 disks diskset "36150" ESM sg8 12 disks diskset "36150"
Enclosure SV21311179 Drawer 4 ESM sg189 12 disks diskset "62228" ESM sg8 12 disks diskset "62228"
Enclosure SV21311179 Drawer 5 ESM sg189 11 disks diskset "05314" ESM sg8 11 disks diskset "05314"
Enclosure SV21311179 sees 58 disks

GSS configuration: 4 enclosures, 6 SSDs, 2 empty slots, 238 disks total, 6 NVRAM partitions

scsi7[15.00.00.00] 0000:11:00.0 [P2 SV21311179 ESM B (sg8)] [P3 SV21419522 ESM A (sg68)]
 [P4 SV21314035 ESM B (sg67)]
scsi8[15.00.00.00] 0000:8b:00.0 [P2 SV21311179 ESM A (sg189)] [P3 SV21313978 ESM B (sg249)]
 [P4 SV21314035 ESM A (sg248)]
scsi9[15.00.00.00] 0000:90:00.0 [P3 SV21313978 ESM A (sg370)] [P4 SV21419522 ESM B (sg431)]
```

Defining the recovery group layout

The left and right recovery groups in a GSS building block without server NVRAM partitions contain a LOG declustered array built on SSDs and two (GSS-24) or three (GSS-26) data declustered arrays (`DA1`, `DA2`, and `DA3`) built on HDDs.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 235

In a GSS 1.5 building block with server NVRAM partitions, there will be an NVR declustered array built on NVRAM partitions, an SSD declustered array built on SSDs, and two or three data declustered arrays (DA1, DA2, and DA3) built on HDDs.

This introduces a new method of storing GNR RAID updates and diagnostics that is both faster and has increased redundancy, and a new set of vdisks for GNR internal logging.

The log tip vdisk will now reside in the faster NVR declustered array.

The SSD declustered array will hold a new log backup vdisk.

The log home vdisk will reside in the first data declustered array (DA1).

A new vdisk of type "log reserved" will be placed in the second and third data declustered arrays. This will be the same size as the log home vdisk, with the intent of making all the data declustered arrays have the same amount of usable space for file system vdisks. (The log reserved vdisks may in the future be used to store additional internal GNR data.)

The NVR declustered array will contain two pdisks. These will be a write-back NVRAM partition from each server, one of which is accessed over the network (rather than through a common SAS fabric).

Since the NVR pdisks are local to each server, recovery groups can no longer be created from the isolated perspective of a single server. The stanza file used to create a recovery group must contain information about the local NVRAM partitions on each of the two servers in a GSS 1.5 building block. The syntax of the **mkrginput** command has been extended to accept two topology files, one from each server in the building block.

To create the recovery group creation stanza files for the left and right recovery groups in a GSS 1.5 building block, provide the names of the topology files for both servers to the **mkrginput** command:

```
# /usr/lpp/mmfs/samples/vdisk/mkrginput server1.top server2.top
```

This will create two stanza files, one for the left recovery group and one for the right. The files will be named after the serial number of the first numbered enclosure in the building block, with an added 'L' or 'R' and a .stanza suffix:

```
# ls *.stanza
SV21314035L.stanza  SV21314035R.stanza
```

The left and right recovery groups must now be created by specifying both servers, using one server as primary for the left and backup for the right, and the other as primary for the right and backup for the left:

```
# mmcrrecoverygroup BB1RGL -F SV21314035L.stanza --servers server1,server2
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
# mmcrrecoverygroup BB1RGR -F SV21314035R.stanza --servers server2,server1
mmcrrecoverygroup: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Verifying recovery group creation

The **mmlsrecoverygroup** command can now be used to show the NVR, SSD, and data declustered arrays:

```
# mmlsrecoverygroup BB1RGL -L

                    declustered
 recovery group       arrays     vdisks  pdisks  format version
 -----------------  -----------  ------  ------  --------------
 BB1RGL                     4       0     121  3.5.0.13
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 236

```
declustered   needs                            replace                scrub       background activity
   array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
-----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
SSD          no            0       3       0          1     558 GiB   14 days  repair-RGD/VCD 10%  low
NVR          no            0       2       0          1    3744 MiB   14 days  repair-RGD/VCD 10%  low
DA1          no            0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low
DA2          no            0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low

                                       declustered                            checksum
vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------

active recovery group server                      servers
------------------------------------------------  -------
server1                                           server1,server2

# mmlsrecoverygroup BB1RGR -L

                    declustered
recovery group        arrays     vdisks  pdisks  format version
-----------------   -----------  ------  ------  --------------
BB1RGR                         4       0     121  3.5.0.13

declustered   needs                            replace                scrub       background activity
   array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
-----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
SSD          no            0       3       0          1     558 GiB   14 days  repair-RGD/VCD 10%  low
NVR          no            0       2       0          1    3744 MiB   14 days  repair-RGD/VCD 10%  low
DA1          no            0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low
DA2          no            0      58       2          2     101 TiB   14 days  repair-RGD/VCD 10%  low

                                       declustered                            checksum
vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------

active recovery group server                      servers
------------------------------------------------  -------
server2                                           server2,server1
```

Defining and creating the vdisks

Each of the left and right recovery groups will now require:

- A log tip vdisk (type `vdiskLogTip`) in the `NVR` declustered array
- A log tip backup vdisk (type `vdiskLogTipBackup`) in the `SSD` declustered array
- A log home vdisk (type `vdiskLog`) in the `DA1` declustered array
- A log reserved vdisk (type `vdiskLogReserved` in the `DA2` declustered array (and in the `DA3` declustered array, in the case of GSS-26)

These will all need to be specified in a vdisk creation stanza file. The vdisk creation example for GSS without NVRAM partitions combined the GNR log vdisks and the file system vdisks in the same stanza file. In the case of GSS 1.5 with NVRAM partitions, it will be simpler to perform vdisk creation in two steps with two separate stanza files, one for the log vdisks and one for the file system vdisks.

The log vdisk creation file for a GSS-24 building block with NVRAM partitions will look like this:

```
# cat mmcrvdisklog.BB1
%vdisk:
  vdiskName=BB1RGLLOGTIP
  rg=BB1RGL
  daName=NVR
  blocksize=2m
  size=48m
  raidCode=2WayReplication
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 237

```
|    diskUsage=vdiskLogTip
|
|  %vdisk:
|    vdiskName=BB1RGLLOGTIPBACKUP
|    rg=BB1RGL
|    daName=SSD
|    blocksize=2m
|    size=48m
|    raidCode=3WayReplication
|    diskUsage=vdiskLogTipBackup
|
|  %vdisk:
|    vdiskName=BB1RGLLOGHOME
|    rg=BB1RGL
|    daName=DA1
|    blocksize=2m
|    size=20g
|    raidCode=4WayReplication
|    diskUsage=vdiskLog
|    longTermEventLogSize=4m
|    shortTermEventLogSize=4m
|    fastWriteLogPct=90
|
|  %vdisk:
|    vdiskName=BB1RGLDA2RESERVED
|    rg=BB1RGL
|    daName=DA2
|    blocksize=2m
|    size=20g
|    raidCode=4WayReplication
|    diskUsage=vdiskLogReserved
|
|  %vdisk:
|    vdiskName=BB1RGRLOGTIP
|    rg=BB1RGR
|    daName=NVR
|    blocksize=2m
|    size=48m
|    raidCode=2WayReplication
|    diskUsage=vdiskLogTip
|
|  %vdisk:
|    vdiskName=BB1RGRLOGTIPBACKUP
|    rg=BB1RGR
|    daName=SSD
|    blocksize=2m
|    size=48m
|    raidCode=3WayReplication
|    diskUsage=vdiskLogTipBackup
|
|  %vdisk:
|    vdiskName=BB1RGRLOGHOME
|    rg=BB1RGR
|    daName=DA1
|    blocksize=2m
|    size=20g
|    raidCode=4WayReplication
|    diskUsage=vdiskLog
|    longTermEventLogSize=4m
|    shortTermEventLogSize=4m
|    fastWriteLogPct=90
|
|  %vdisk:
|    vdiskName=BB1RGRDA2RESERVED
|    rg=BB1RGR
|    daName=DA2
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 238

```
  blocksize=2m
  size=20g
  raidCode=4WayReplication
  diskUsage=vdiskLogReserved
```

The parameters chosen for `size`, `blocksize`, `raidCode`, `fastWriteLogPct`, and the event log sizes are standard and have been carefully calculated, and they should not be changed. The only difference in the vdisk log stanza files between two building blocks will be in the recovery group and vdisk names. (In the case of a GSS-26 building block with NVRAM partitions, there will be an additional `vdiskLogReserved` for `DA3`, with parameters otherwise identical to the `DA2` log reserved vdisk.)

The log vdisks for the example GSS 1.5 building block `BB1` can now be created with the **mmcrvdisk** command:

```
# mmcrvdisk -F mmcrvdisklog.BB1
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGTIPBACKUP
mmcrvdisk: [I] Processing vdisk BB1RGLLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGLDA2RESERVED
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGTIPBACKUP
mmcrvdisk: [I] Processing vdisk BB1RGRLOGHOME
mmcrvdisk: [I] Processing vdisk BB1RGRDA2RESERVED
mmcrvdisk: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Creation of the GSS 1.5 log vdisks may be verified through the **mmlsvdisk** and **mmlsrecoverygroup** commands:

```
# mmlsvdisk
                                                    declustered  block size
 vdisk name          RAID code        recovery group         array       in KiB    remarks
 ------------------  ---------------  ------------------  -----------  ----------  -------
 BB1RGLDA2RESERVED   4WayReplication  BB1RGL              DA2                2048  logRsvd
 BB1RGLLOGHOME       4WayReplication  BB1RGL              DA1                2048  log
 BB1RGLLOGTIP        2WayReplication  BB1RGL              NVR                2048  logTip
 BB1RGLLOGTIPBACKUP  3WayReplication  BB1RGL              SSD                2048  logTipBackup
 BB1RGRDA2RESERVED   4WayReplication  BB1RGR              DA2                2048  logRsvd
 BB1RGRLOGHOME       4WayReplication  BB1RGR              DA1                2048  log
 BB1RGRLOGTIP        2WayReplication  BB1RGR              NVR                2048  logTip
 BB1RGRLOGTIPBACKUP  3WayReplication  BB1RGR              SSD                2048  logTipBackup

# mmlsrecoverygroup BB1RGL -L

                    declustered
 recovery group       arrays     vdisks  pdisks  format version
 -----------------  -----------  ------  ------  --------------
 BB1RGL                      4       4     121  3.5.0.13

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 SSD          no            1       3       0          1     558 GiB   14 days  scrub        4%  low
 NVR          no            1       2       0          1    3648 MiB   14 days  scrub        4%  low
 DA1          no            1      58       2          2     101 TiB   14 days  scrub        0%  low
 DA2          no            1      58       2          2     101 TiB   14 days  scrub        0%  low

                                      declustered                            checksum
 vdisk               RAID code             array     vdisk size  block size  granularity  state remarks
 ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------
 BB1RGLLOGTIP        2WayReplication     NVR              48 MiB      2 MiB       512      ok    logTip
 BB1RGLLOGTIPBACKUP  3WayReplication     SSD              48 MiB      2 MiB       512      ok    logTipBackup
 BB1RGLLOGHOME       4WayReplication     DA1              20 GiB      2 MiB       512      ok    log
 BB1RGLDA2RESERVED   4WayReplication     DA2              20 GiB      2 MiB       512      ok    logRsvd

 active recovery group server                    servers
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 239

```
|  ----------------------------------------------  -------
|  server1                                         server1,server2
|
| # mmlsrecoverygroup BB1RGR -L
|
|                     declustered
|  recovery group       arrays     vdisks  pdisks  format version
|  -----------------  -----------  ------  ------  --------------
|  BB1RGR                      4       4     121  3.5.0.13
|
|  declustered   needs                              replace               scrub       background activity
|     array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
|  -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
|  SSD          no             1       3       0          1     558 GiB   14 days  scrub        4%  low
|  NVR          no             1       2       0          1    3648 MiB   14 days  scrub        4%  low
|  DA1          no             1      58       2          2     101 TiB   14 days  scrub        1%  low
|  DA2          no             1      58       2          2     101 TiB   14 days  scrub        1%  low
|
|                                      declustered                              checksum
|  vdisk               RAID code              array     vdisk size  block size  granularity  state remarks
|  ------------------  ------------------  -----------  ----------  ----------  -----------  ----- -------
|  BB1RGRLOGTIP        2WayReplication     NVR              48 MiB      2 MiB       512      ok    logTip
|  BB1RGRLOGTIPBACKUP  3WayReplication     SSD              48 MiB      2 MiB       512      ok    logTipBackup
|  BB1RGRLOGHOME       4WayReplication     DA1              20 GiB      2 MiB       512      ok    log
|  BB1RGRDA2RESERVED   4WayReplication     DA2              20 GiB      2 MiB       512      ok    logRsvd
|
|  active recovery group server                    servers
|  -----------------------------------------------  -------
|  server2                                         server2,server1
```

The creation of file system vdisks, NSDs, and of a file system can now proceed just as in the example for GSS without NVRAM partitions, by omitting the log vdisks from the stanza file and including only the desired file system vdisks.

Example scenario: Replacing failed disks in a GSS recovery group

The scenario presented here shows how to detect and replace failed disks in a recovery group built on a GSS building block.

Note: When replacing a failed disk, the **mmchcarrier** command will attempt to install the latest firmware on the disk by invoking the sample callback script `/usr/lpp/mmfs/samples/vdisk/tspreparenewpdiskforuse` (if present). To use this callback script, copy it to `/usr/lpp/mmfs/bin/tspreparenewpdiskforuse` and make it executable.

Detecting failed disks in your GSS enclosure

Assume a GSS-24 building block on which the following two recovery groups are defined:

- `BB1RGL`, containing the disks in the left side of each drawer
- `BB1RGR`, containing the disks in the right side of each drawer

Each recovery group contains the following:

- one log declustered array (`LOG`)
- two data declustered arrays (`DA1`, `DA2`)

The data declustered arrays are defined according to GSS-24 best practice as follows:

- 58 pdisks per data declustered array
- default disk replacement threshold value set to 2

The replacement threshold of 2 means that GPFS Native RAID will only require disk replacement when two or more disks have failed in the declustered array; otherwise, rebuilding onto spare space or

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 240

reconstruction from redundancy will be used to supply affected data. This configuration can be seen in the output of **mmlsrecoverygroup** for the recovery groups, shown here for BB1RGL:

```
# mmlsrecoverygroup BB1RGL -L

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 BB1RGL                    3       5     119

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 LOG          no            1       3       0          1     534 GiB   14 days  scrub        1%  low
 DA1          yes           2      58       2          2       0   B   14 days  rebuild-1r   4%  low
 DA2          no            2      58       2          2    1024 MiB   14 days  scrub       27%  low

                                     declustered                              checksum
 vdisk               RAID code          array     vdisk size  block size  granularity  remarks
 ------------------  -----------------  -----------  ----------  ----------  -----------  -------
 BB1RGLLOG           3WayReplication    LOG            8192 MiB    256 KiB       512      log
 BB1RGLMETA1         4WayReplication    DA1            1024 GiB      1 MiB     32 KiB
 BB1RGLDATA1         8+3p               DA1              70 TiB     16 MiB     32 KiB
 BB1RGLMETA2         4WayReplication    DA2            1024 GiB      1 MiB     32 KiB
 BB1RGLDATA2         8+3p               DA2              70 TiB     16 MiB     32 KiB

 active recovery group server                    servers
 ----------------------------------------------  -------
 server1                                         server1,server2
```

The indication that disk replacement is called for in this recovery group is the value of yes in the needs service column for declustered array DA1.

The fact that DA1 is undergoing rebuild of its RAID tracks that can tolerate one strip failure is by itself not an indication that disk replacement is required; it merely indicates that data from a failed disk is being rebuilt onto spare space. Only if the replacement threshold has been met will disks be marked for replacement and the declustered array marked as needing service.

GPFS Native RAID provides several indications that disk replacement is required:

- entries in the Linux syslog
- the GPFS **pdReplacePdisk** callback, which can be configured to run an administrator-supplied script at the moment a pdisk is marked for replacement
- the output from the following commands, which may be performed from the command line on any GPFS cluster node (see the examples that follow):
  1. **mmlsrecoverygroup** with the **-L** flag shows yes in the needs service column
  2. **mmlsrecoverygroup** with the **-L** and **--pdisk** flags; this shows the states of all pdisks, which may be examined for the replace pdisk state
  3. **mmlspdisk** with the **--replace** flag, which lists only those pdisks that are marked for replacement

Note: Because the output of **mmlsrecoverygroup -L --pdisk** is very long, this example shows only some of the pdisks (but includes those marked for replacement).

```
# mmlsrecoverygroup BB1RGL -L --pdisk

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 BB1RGL                    3       5     119

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 LOG          no            1       3       0          1     534 GiB   14 days  scrub        1%  low
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 241

```
 DA1          yes           2      58       2          2       0   B   14 days  rebuild-1r   4%  low
 DA2          no            2      58       2          2    1024 MiB   14 days  scrub       27%  low

                    number of    declustered
 pdisk              active paths    array     free space  state
 -----------------  ------------  -----------  ----------  -----
 [...]
 e1d4s06                 2        DA1              62 GiB  ok
 e1d5s01                 0        DA1              70 GiB  slow/noPath/systemDrain/noRGD/noVCD/replace
 e1d5s02                 2        DA1              64 GiB  ok
 e1d5s03                 2        DA1              63 GiB  ok
 e1d5s04                 0        DA1              64 GiB  failing/noPath/systemDrain/noRGD/noVCD/replace
 e1d5s05                 2        DA1              63 GiB  ok
 [...]
```

The preceding output shows that the following pdisks are marked for replacement:

- e1d5s01 in DA1
- e1d5s04 in DA1

The naming convention used during recovery group creation indicates that these are the disks in Enclosure 1 Drawer 5 Slot 1 and Enclosure 1 Drawer 5 Slot 4. To confirm the physical locations of the failed disks, use the **mmlspdisk** command to list information about those pdisks in declustered array DA1 of recovery group BB1RGL that are marked for replacement:

```
# mmlspdisk BB1RGL --declustered-array DA1 --replace
pdisk:
   replacementPriority = 0.98
   name = "e1d5s01"
   device = ""
   recoveryGroup = "BB1RGL"
   declusteredArray = "DA1"
   state = "slow/noPath/systemDrain/noRGD/noVCD/replace"
   freeSpace = 75161927680
   fru = "90Y8597"
   location = "SV21314035-5-1"
   WWN = "naa.5000CCA01B510EFC"
   server = "server1"
   reads = 32750112
   writes = 53114318
   bytesReadInGiB = 15585.770
   bytesWrittenInGiB = 25117.229
   IOErrors = 0
   IOTimeouts = 41
   mediaErrors = 0
   checksumErrors = 0
   pathErrors = 0
   relativePerformance = 0.661
   dataBadness = 0.000
   rgIndex = 26
   userLocation = "Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 1"
pdisk:
   replacementPriority = 0.98
   name = "e1d5s04"
   device = ""
   recoveryGroup = "BB1RGL"
   declusteredArray = "DA1"
   state = "failing/noPath/systemDrain/noRGD/noVCD/replace"
   freeSpace = 68719476736
   fru = "90Y8597"
   location = "SV21314035-5-4"
   WWN = "naa.5000CCA01B4B702C"
   server = "server1"
   reads = 51176754
   writes = 80568342
   bytesReadInGiB = 24385.029
   bytesWrittenInGiB = 38243.908
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 242

```
   IOErrors = 11
   IOTimeouts = 422
   mediaErrors = 0
   checksumErrors = 0
   pathErrors = 3
   relativePerformance = 0.624
   dataBadness = 0.000
   rgIndex = 29
   userLocation = "Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 4"
```

The physical locations of the failed disks are confirmed to be consistent with the pdisk naming convention and with the GPFS Native RAID component database:

```
------------------------------------------------------------------------------------
Disk          Location         User Location
------------------------------------------------------------------------------------
pdisk e1d5s01  SV21314035-5-1  Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 1
------------------------------------------------------------------------------------
pdisk e1d5s04  SV21314035-5-4  Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 4
------------------------------------------------------------------------------------
```

This shows how the component database provides an easier-to-use location reference for the affected physical disks. The pdisk name `e1d5s01` signifies very simply "Enclosure 1 Drawer 5 Slot 1." The location additionally provides the serial number of enclosure 1, `SV21314035`, with the drawer and slot number. But the user location that has been defined in the component database can be used to precisely locate the disk in a given equipment rack and a named disk enclosure: This is the disk enclosure labelled "`BB1ENC1`," found in compartments `U01 - U04` of the rack labelled "`BB1`," and the disk is in drawer 5, slot 1 of that enclosure.

The relationship between the enclosure serial number and the user location can be seen with the **mmlscomp** command:

```
# mmlscomp --serial-number SV21314035

    Storage Enclosure Components

Comp ID  Part Number  Serial Number  Name     Display ID
-------  -----------  -------------  -------  ----------
      2  181880E      SV21314035     BB1ENC1
```

Replacing failed disks in a GSS-24 recovery group

Note: In this example, it is assumed that two new disks with the appropriate Field Replaceable Unit (FRU) code, as indicated by the fru attribute (`90Y8597` in this case), have been obtained as replacements for the failed pdisks `e1d5s01` and `e1d5s04`.

Replacing each disk is a three-step process:

1. Using the **mmchcarrier** command with the **--release** flag to inform GPFS to locate the disk, suspend it, and allow it to be removed.
2. Locating and removing the failed disk and replacing it with a new one.
3. Using the **mmchcarrier** command with the **--replace** flag to begin use of the new disk.

GPFS Native RAID assigns a priority to pdisk replacement. Disks with smaller values for the **replacementPriority** attribute should be replaced first. In this example, the only failed disks are in `DA1` and both have the same **replacementPriority**.

Disk `e1d5s01` is chosen to be replaced first.

1. To release pdisk `e1d5s01` in recovery group `BB1RGL`:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 243

```
# mmchcarrier BB1RGL --release --pdisk e1d5s01
[I] Suspending pdisk e1d5s01 of RG BB1RGL in location SV21314035-5-1.
[I] Location SV21314035-5-1 is Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 1.
[I] Carrier released.

  - Remove carrier.
  - Replace disk in location SV21314035-5-1 with FRU 90Y8597.
  - Reinsert carrier.
  - Issue the following command:

      mmchcarrier BB1RGL --replace --pdisk 'e1d5s01'

    Repair timer is running.  Perform the above within 5 minutes
    to avoid pdisks being reported as missing.
```

GPFS Native RAID issues instructions as to the physical actions that must be taken, and repeats the user-defined location to assist in finding the disk. As disk replacement in a GSS-24 building block affects only the disk being replaced, the five-minute warning serves mostly as a reminder to act promptly.

2. To allow the enclosure `BB1ENC1` with serial number `SV21314035` to be located and identified, GPFS Native RAID will turn on the enclosure's amber "service required" LED. The enclosure's bezel should be removed. This will reveal that the amber "service required" and blue "service allowed" LEDs for drawer 5 have been turned on.

   Drawer 5 should then be unlatched and pulled open. The disk in slot 1 will be seen to have its amber and blue LEDs turned on.

   Unlatch and pull up the handle for the identified disk in slot 1. Lift the failed disk out and set it aside. The drive LEDs will turn off when the slot is empty. A new disk with FRU `90Y8597` should be lowered in place and have its handle pushed down and latched.

   Since the second disk replacement in this example is also in drawer 5 of the same enclosure, leave the drawer open and the enclosure bezel off. If the next replacement were in a different drawer, the drawer would be closed; and if the next replacement were in a different enclosure, the enclosure bezel would be replaced.

3. To finish the replacement of pdisk `e1d5s01`:

   ```
   # mmchcarrier BB1RGL --replace --pdisk e1d5s01
   [I] The following pdisks will be formatted on node server1:
       /dev/sdmi
   [I] Pdisk e1d5s01 of RG BB1RGL successfully replaced.
   [I] Resuming pdisk e1d5s01#026 of RG BB1RGL.
   [I] Carrier resumed.
   ```

   When the **mmchcarrier --replace** command returns successfully, GPFS Native RAID will begin rebuilding and rebalancing RAID strips onto the new disk, which has assumed the pdisk name `e1d5s01`. The failed pdisk may remain in a temporary form (indicated here by the name `e1d5s01#026`) until all data from it has been rebuilt, at which point it is finally deleted. Notice that only one block device name is mentioned as being formatted as a pdisk; the second path will be discovered in the background.

   Disk `e1d5s04` will still be marked for replacement, and `DA1` of `BBRGL` will still need service. This is because GPFS Native RAID replacement policy expects all failed disks in the declustered array to be replaced once the replacement threshold is reached.

Pdisk `e1d5s04` is then replaced following the same process.

1. Release pdisk `e1d5s04` in recovery group `BB1RGL`:

   ```
   # mmchcarrier BB1RGL --release --pdisk e1d5s04
   [I] Suspending pdisk e1d5s04 of RG BB1RGL in location SV21314035-5-4.
   [I] Location SV21314035-5-4 is Rack BB1 U01-04, Enclosure BB1ENC1 Drawer 5 Slot 4.
   [I] Carrier released.

     - Remove carrier.
     - Replace disk in location SV21314035-5-4 with FRU 90Y8597.
     - Reinsert carrier.
   ```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 244

```
- Issue the following command:

    mmchcarrier BB1RGL --replace --pdisk 'e1d5s04'

  Repair timer is running.  Perform the above within 5 minutes
  to avoid pdisks being reported as missing.
```

2. Find the enclosure and drawer, unlatch and remove the disk in slot 4, place a new disk in slot 4, push in the drawer, and replace the enclosure bezel.
3. To finish the replacement of pdisk e1d5s04:

```
# mmchcarrier BB1RGL --replace --pdisk e1d5s04
[I] The following pdisks will be formatted on node server1:
    /dev/sdfd
[I] Pdisk e1d5s04 of RG BB1RGL successfully replaced.
[I] Resuming pdisk e1d5s04#029 of RG BB1RGL.
[I] Carrier resumed.
```

The disk replacements may be confirmed with **mmlsrecoverygroup -L --pdisk**:

```
# mmlsrecoverygroup BB1RGL -L --pdisk

                   declustered
 recovery group       arrays     vdisks  pdisks
 -----------------  -----------  ------  ------
 BB1RGL                     3       5     121

 declustered   needs                            replace                scrub       background activity
    array     service  vdisks  pdisks  spares  threshold  free space  duration  task   progress  priority
 -----------  -------  ------  ------  ------  ---------  ----------  --------  -------------------------
 LOG          no            1       3       0          1     534 GiB   14 days  scrub        1%  low
 DA1          no            2      60       2          2    3647 GiB   14 days  rebuild-1r   4%  low
 DA2          no            2      58       2          2    1024 MiB   14 days  scrub       27%  low

                    number of    declustered
 pdisk              active paths     array     free space  state
 -----------------  ------------  -----------  ----------  -----
 [...]
 e1d4s06                 2        DA1             62 GiB  ok
 e1d5s01                 2        DA1           1843 GiB  ok
 e1d5s01#026             0        DA1             70 GiB  slow/noPath/systemDrain/adminDrain/noRGD/noVCD
 e1d5s02                 2        DA1             64 GiB  ok
 e1d5s03                 2        DA1             63 GiB  ok
 e1d5s04                 2        DA1           1853 GiB  ok
 e1d5s04#029             0        DA1             64 GiB  failing/noPath/systemDrain/adminDrain/noRGD/noVCD
 e1d5s05                 2        DA1             62 GiB  ok
 [...]
```

Notice that the temporary pdisks (e1d5s01#026 and e1d5s04#029) representing the now-removed physical disks are counted toward the total number of pdisks in the recovery group BB1RGL and the declustered array DA1. They will exist until RAID rebuild completes the reconstruction of the data that they carried onto other disks (including their replacements). When rebuild completes, the temporary pdisks will disappear, and the number of disks in DA1 will once again be 58, and the number of disks in BBRGL will once again be 119.

Example scenario: Replacing failed GSS storage enclosure components

The scenario presented here shows how to detect and replace failed storage enclosure components in a GSS building block.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 245

Detecting failed storage enclosure components

The **mmlsenclosure** command can be used to show you which enclosures need service along with the specific component. A best practice is to run this command each day to check for failures.

```
# mmlsenclosure all -L --not-ok

                   needs  nodes
 serial number     service
 -------------     ------- ------
 SV21313971        yes     c45f02n01-ib0.gpfs.net

 component type  serial number     component id    failed value   unit   properties
 --------------  -------------     ------------    ------ -----   ----   ----------
 fan             SV21313971        1_BOT_LEFT      yes            RPM    FAILED
```

This indicates that enclosure SV21313971 has a failed fan. Refer to *IBM GPFS Storage Server Hardware Installation and Service Guide*.

Once you are ready to replace the failed component, use the **mmchenclosure** command to identify whether it is safe to complete the repair action or whether GPFS needs to be shut down first:

```
# mmchenclosure SV21313971 --component fan --component-id 1_BOT_LEFT

mmenclosure: Proceed with the replace operation.
```

The fan can now be replaced.

Special note about detecting failed enclosure components

In the following example, only the enclosure itself is being called out as having failed; the specific component that has actually failed is not identified. This typically means that there are drive "Service Action Required (Fault)" LEDs that have been turned on in the drawers. In such a situation, the **mmlspdisk all --not-ok** command can be used to check for dead or failing disks.

```
mmlsenclosure all -L --not-ok

                   needs  nodes
 serial number     service
 -------------     ------- ------
 SV13306129        yes     c45f01n01-ib0.gpfs.net

 component type  serial number     component id    failed value   unit   properties
 --------------  -------------     ------------    ------ -----   ----   ----------
 enclosure       SV13306129        ONLY            yes                    NOT_IDENTIFYING,FAILED
```

Example scenario: Replacing a failed GSS storage enclosure drawer

The scenario presented here shows how to use the **chdrawer** sample script as a service aid in replacing a GSS enclosure drawer while GPFS is active.

The **chdrawer** sample script is intended to be used in GSS-26 configurations where the declustered arrays have been laid out across all the drawers so that removing a drawer affects only two disks in each declustered array.

Notes:

1. In this layout, follow these recommendations:
   - Run 8+3p or 4-way mirroring only
   - Always have two spares per declustered array.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 246

You can use 8+2p or 3-way mirroring, but this means that once the drawer is removed, every declustered array will be running in critical mode. Therefore, one more sector error or disk failure will result in data being unavailable or worse.

2. A drawer can be replaced in a GSS-24 configuration laid out such that removing a drawer affects three disks in each declustered array, but *only* in the case where the vdisks were using 8+3p or 4-way mirroring.

The syntax of the **chdrawer** sample script is:

```
chdrawer EnclosureSerialNumber DrawerNumber
     {--release [--force-release] | --replace [--force-replace]} [--dry-run]
```

where:

EnclosureSerialNumber
Specifies the enclosure serial number, as displayed by **mmlsenclosure**.

DrawerNumber
Specifies the drawer number to be replaced. Drawers are numbered from top to bottom in an enclosure.

--release
Prepares the drawer for disk replacement by draining all the pdisks.

--replace
Resumes all the pdisks once the drawer has been replaced (with all the former pdisks in their corresponding slots).

--dry-run
Runs the command without actually changing the pdisk states; checks whether there is enough redundancy to safely replace the drawer on a live system.

--force-release
Forces the **--release** option. Changes the pdisk states to `serviceDrain` despite warnings.

--force-replace
Forces the **--replace** option. Ends the service drain state despite warnings.

To replace drawer `3` in enclosure `SV21106537`, follow these steps:

1. Start the release sequence using the following command:
   ```
   /usr/lpp/mmfs/samples/vdisk/chdrawer SV21106537 3 --release
   ```
   The command performs the following actions:
   a. Discovers all the pdisks that are in the drawer.
   b. Checks that each declustered array has two spares.
   c. Checks that there are no other disks down that might prevent a successful rebuild.
   d. Places the pdisks in `serviceDrain` state, which will drain all the data from the affected pdisks.
   e. Waits until all the data is drained.
2. Record the location of the drives and label them, as you will later need to replace them in the corresponding slots of the new drawer.
3. Disconnect the drawer, repair it, and put it back in. (See *IBM GPFS Storage Server Hardware Installation and Service Guide* for physical replacement instructions.)
4. Complete the replacement sequence using the following command:
   ```
   /usr/lpp/mmfs/samples/vdisk/chdrawer SV21106537 3 --replace
   ```
   The command performs the following actions:
   a. Verifies that the expected number of pdisks have been found.
   b. Removes the `serviceDrain` state, which will start rebalancing data back onto the pdisks.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 247

Example scenario: Replacing a failed GSS storage enclosure

The scenario presented here shows how to replace a failed GSS storage enclosure.

To replace a filed storage enclosure, follow these steps:

1. Shut down GPFS across the cluster by issuing the following command:

   ```
   mmshutdown -a
   ```

2. Record the drawer and slot locations of the drives and label them, as you will need to move them to the corresponding slots of the new drawers of the new enclosure.
3. Remove the SAS connections in the rear of the enclosure.
4. Remove the enclosure.
5. Install the new enclosure.
6. Transfer the disks to the new drawers of the new enclosure, referring to the information saved in step 2.
7. Connect the SAS connections in the rear of the new enclosure.
8. Power up the enclosure.
9. Verify the SAS topology on the servers to ensure that all drives from the new storage enclosure are present.
10. Update the necessary firmware on the new storage enclosure as needed.
11. Startup GPFS across the cluster

    ```
    mmstartup -a
    ```

Example scenario: Checking the health of a GSS configuration

The scenario presented here shows how to use the **gnrhealthcheck** sample script to check the general health of a GSS configuration.

The syntax of the **gnrhealthcheck** sample script is:

```
gnrhealthcheck [--topology] [--enclosure] [--rg] [--pdisk]
               [--perf-dd] [--local]
```

where:

--topology
Checks the operating system topology. Runs **mmgetpdisktopology** and **topsummary** to look for cabling and path issues.

--enclosure
Checks enclosures. Runs **mmlsenclosure** to look for failures.

--rg
Checks recovery groups. Runs **mmlsrecoverygroup** to check whether all recovery groups are active and whether the active server is the primary server. Also checks for any recovery groups that need service.

--pdisk
Checks pdisks. Runs **mmlspdisk** to check that each pdisk has two paths.

--perf-dd
Checks basic performance of disks. Runs a dd read to each potential GNR disk drive for a GB and reports basic performance stats. Reads are done six disks at a time. These statistics will only be meaningful if run on an idle system. Available on Linux only.

--local
Runs tests only on the invoking node.

The default is to check everything *except* **--perf-dd** arguments on all NSD server nodes.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 248

Output return codes:

0 No problems were found.

1 Problems were found, and information was displayed.

Note: The default is to display to stdout. There may be a large amount of data, so it is recommended that you pipe the output to a file.

Examples

1. To run a health check on the local server nodes and place output in /tmp/gnrhealthcheck.out, issue the following command:

```
gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
```

In this example all checks are successful.

The system displays information similar to this:

```
############################################################
# Beginning topology checks.
############################################################
Topology checks successful.

############################################################
# Beginning enclosure checks.
############################################################
Enclosure checks successful.

############################################################
# Beginning recovery group checks.
############################################################
Recovery group checks successful.

############################################################
# Beginning pdisk checks.
############################################################

Pdisk group checks successful.
```

2. To run a health check on the local server nodes and place output in /tmp/gnrhealthcheck.out, issue the following command:

```
gnrhealthcheck --local | tee /tmp/gnrhealthcheck.out
```

In this example, there are numerous issues to be investigated.

The system displays information similar to this:

```
############################################################
# Beginning topology checks.
############################################################
Found topology problems on node c45f01n01-ib0.gpfs.net

DCS3700 enclosures found: 0123456789AB SV11812206 SV12616296 SV13306129
Enclosure 0123456789AB (number 1):
Enclosure 0123456789AB ESM A sg244[0379][scsi8 port 4] ESM B sg4[0379][scsi7 port 4]
Enclosure 0123456789AB Drawer 1 ESM sg244 12 disks diskset "19968" ESM sg4 12 disks diskset "19968"
Enclosure 0123456789AB Drawer 2 ESM sg244 12 disks diskset "11294" ESM sg4 12 disks diskset "11294"
Enclosure 0123456789AB Drawer 3 ESM sg244 12 disks diskset "60155" ESM sg4 12 disks diskset "60155"
Enclosure 0123456789AB Drawer 4 ESM sg244 12 disks diskset "03345" ESM sg4 12 disks diskset "03345"
Enclosure 0123456789AB Drawer 5 ESM sg244 11 disks diskset "33625" ESM sg4 11 disks diskset "33625"
Enclosure 0123456789AB sees 59 disks

Enclosure SV12616296 (number 2):
Enclosure SV12616296 ESM A sg63[0379][scsi7 port 3] ESM B sg3[0379][scsi9 port 4]
Enclosure SV12616296 Drawer 1 ESM sg63 11 disks diskset "51519" ESM sg3 11 disks diskset "51519"
Enclosure SV12616296 Drawer 2 ESM sg63 12 disks diskset "36246" ESM sg3 12 disks diskset "36246"
Enclosure SV12616296 Drawer 3 ESM sg63 12 disks diskset "53750" ESM sg3 12 disks diskset "53750"
Enclosure SV12616296 Drawer 4 ESM sg63 12 disks diskset "07471" ESM sg3 12 disks diskset "07471"
Enclosure SV12616296 Drawer 5 ESM sg63 11 disks diskset "16033" ESM sg3 11 disks diskset "16033"
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 249

```
Enclosure SV12616296 sees 58 disks

Enclosure SV11812206 (number 3):
Enclosure SV11812206 ESM A sg66[0379][scsi9 port 3] ESM B sg6[0379][scsi8 port 3]
Enclosure SV11812206 Drawer 1 ESM sg66 11 disks diskset "23334" ESM sg6 11 disks diskset "23334"
Enclosure SV11812206 Drawer 2 ESM sg66 12 disks diskset "16332" ESM sg6 12 disks diskset "16332"
Enclosure SV11812206 Drawer 3 ESM sg66 12 disks diskset "52806" ESM sg6 12 disks diskset "52806"
Enclosure SV11812206 Drawer 4 ESM sg66 12 disks diskset "28492" ESM sg6 12 disks diskset "28492"
Enclosure SV11812206 Drawer 5 ESM sg66 11 disks diskset "24964" ESM sg6 11 disks diskset "24964"
Enclosure SV11812206 sees 58 disks

Enclosure SV13306129 (number 4):
Enclosure SV13306129 ESM A sg64[0379][scsi8 port 2] ESM B sg353[0379][scsi7 port 2]
Enclosure SV13306129 Drawer 1 ESM sg64 11 disks diskset "47887" ESM sg353 11 disks diskset "47887"
Enclosure SV13306129 Drawer 2 ESM sg64 12 disks diskset "53906" ESM sg353 12 disks diskset "53906"
Enclosure SV13306129 Drawer 3 ESM sg64 12 disks diskset "35322" ESM sg353 12 disks diskset "35322"
Enclosure SV13306129 Drawer 4 ESM sg64 12 disks diskset "37055" ESM sg353 12 disks diskset "37055"
Enclosure SV13306129 Drawer 5 ESM sg64 11 disks diskset "16025" ESM sg353 11 disks diskset "16025"
Enclosure SV13306129 sees 58 disks

DCS3700 configuration: 4 enclosures, 1 SSD, 7 empty slots, 233 disks total
Location 0123456789AB-5-12 appears empty but should have an SSD
Location SV12616296-1-3 appears empty but should have an SSD
Location SV12616296-5-12 appears empty but should have an SSD
Location SV11812206-1-3 appears empty but should have an SSD
Location SV11812206-5-12 appears empty but should have an SSD

scsi7[07.00.00.00] 0000:11:00.0 [P2 SV13306129 ESM B (sg353)] [P3 SV12616296 ESM A (sg63)] [P4 0123456789AB ESM B (sg4)]
scsi8[07.00.00.00] 0000:8b:00.0 [P2 SV13306129 ESM A (sg64)] [P3 SV11812206 ESM B (sg6)] [P4 0123456789AB ESM A (sg244)]
scsi9[07.00.00.00] 0000:90:00.0 [P3 SV11812206 ESM A (sg66)] [P4 SV12616296 ESM B (sg3)]

############################################################
# Beginning enclosure checks.
############################################################
Enclosure checks successful.

############################################################
# Beginning recovery group checks.
############################################################
Found recovery group BB1RGR, primary server is not the active server.

############################################################
# Beginning pdisk checks.
############################################################
Found recovery group BB1RGL pdisk e4d5s06 has 0 paths.
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 250

Chapter 13. Active file management

Active file management (AFM) is a scalable, high-performance, file system caching layer integrated with the GPFS cluster file system. AFM allows you to create associations from a local GPFS cluster to a remote cluster or storage, and to define the location and flow of file data to automate the management of the data. This allows you to implement a single namespace view across sites around the world.

Note: This feature is available with GPFS Standard Edition or higher.

AFM masks wide-area network latencies and outages by using GPFS to cache massive data sets, allowing data access and modifications even when remote storage cluster is unavailable. In addition, AFM performs updates to the remote cluster asynchronously, which allows applications to continue operating while not being constrained by limited outgoing network bandwidth.

The AFM implementation leverages the inherent scalability of GPFS to provide a multinode, consistent cache of data located at a home cluster. By integrating with the file system, AFM provides a POSIX-compliant interface, making the cache completely transparent to applications. AFM is easy to deploy, as it relies on open standards for high-performance file serving and does not require any proprietary hardware or software to be installed at the home cluster.

By using NFSv3 or GPFS protocol to cache data, AFM can improve network performance to any home cluster.

Note: Encryption cannot be applied to AFM-managed filesets.

Active file management architecture

AFM uses a home-and-cache model in which a single home provides the primary storage of data, and exported data is cached in a local GPFS file system.

Home Home can be an NFS export from a remote cluster. The export point can be a local file system in the remote cluster or a GPFS file system or a GPFS fileset in the remote cluster. AFM uses a proprietary protocol over NFS.

In addition, AFM is supported when a remote file system is mounted on the cache cluster using GPFS protocol. Native GPFS protocol utilizes remote file system mount based over a multicluster configuration to function as the AFM target. This requires that a multicluster setup exist between the home and cache before AFM can use the home cluster's file system mount on the remote cluster for AFM operations.

Architecturally, AFM works with any file system at the home cluster; however, ACLs, extended attributes, and sparse files are only supported when the home file system is GPFS, irrespective of whether NFS or GPFS target is used. The **mmafmconfig** command should be run on the home cluster to enable this support.

Cache The container used to cache home data is a GPFS fileset. Each AFM-enabled fileset has a single home cluster associated with it (represented by the hostname of the home server).

Each cache fileset in a cluster is served by one of the nodes designated as gateway in the cluster. The gateway node mapped to serve a fileset is called the metadata server (MDS) of the fileset. The MDS acts as the owner for the fileset. All other nodes in the cluster, including other gateways, become application nodes for the fileset. A fileset can have multiple application nodes (which service application data requests). All other gateway nodes can be configured to help the MDS in fetching data to and from the home cluster. See the Parallel I/O section of the guide.

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 251

The split between application and gateway nodes is conceptual, and any node in the cache cluster can function as both a gateway node and an application node based on its configuration. The gateway nodes can be viewed as the edge of the cache cluster that can communicate with the home cluster, while the application nodes interface with the application. Gateway and application nodes communicate with each other via internal RPC requests.

AFM is supported on AIX and Linux nodes. Only Linux nodes can be assigned as gateway nodes.

Any cluster can be a home cluster, a cache cluster, or both. In typical usage, a home would be in one GPFS cluster and a cache would be defined in another, but this is not required. In fact, a cluster can be a home for one fileset and can be a cache for another fileset; and multiple AFM-enabled filesets may be defined in one cache, each caching from a different home cluster. This provides great flexibility in how you can leverage the caching behavior.

A cache can request data by reading a file or by pre-fetching the data. Any time a file is read, if the file data is not yet in the cache or is not up to date, the data is copied from the home into the cache.

Multiple cache filesets can read data from a single home. In single-writer mode, only one cache can write data to a single home. In independent-writer (IW) mode, multiple caches can write to a single home, as long as each cache writes to different files. In case multiple caches write to same file, the sequence of updates is nondeterministic.

AFM filesets can cache extended attributes and ACLs. In order to enable this functionality, the home needs to issue the **mmafmconfig** command.

Notes:

1. AFM uses certain internal directories, such as the following, which should not be altered or removed:
   - `.afm` at home
   - `.ptrash` (see "Failover of cache filesets" on page 240
   - `.pconflicts` in cache (see "Failover of cache filesets" on page 240)
2. User IDs and group IDs should be managed the same way across cache and home.

Caching modes

The following table shows the AFM caching modes that are available to control the flow of data:

Table 39. AFM modes

Mode	Description
Local-update (LU)	Cached data (data from home) is read-only. The difference from read-only mode is that you can create and modify files in the cache fileset. Files that are created or modified in the cache are considered local updates. Local updates are never pushed back to home. Once a file is modified in the cache, the file is no longer compared to the version at home to verify that it is up to date. Appending to, truncating, or writing to an uncached file will fetch the entire file to cache before making the change locally. **Note:** Any small change done in the LU fileset directory could cause the fileset to be marked as local and lose context with the home. For example, running **chmod** or **chown** of the LU fileset root directory will cause the entire fileset to be out of sync with the home.
Read-only (RO)	Data in the cache is read-only. Creating or modifying files in the cache fileset is not allowed.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 252

Table 39. AFM modes (continued)

Mode	Description
Single-writer (SW)	One cache fileset does all the writing. The assumption is that all of the other cache filesets associated with the same home are configured as read-only or local-update. This mode avoids possible write conflicts. There is no explicit enforcement of a single-writer mode in GPFS. Rather, the administrator needs to guarantee that there are no other writeable caches associated with this home and that no writes are done at home. On single writer filesets, a write at home is not allowed. However, if there is a write/corruption at home, inadvertently or due to some error situation, it results in a conflicted scenario for the single-writer fileset. The administrator-monitored command **resync** can be used to recover home. When running **resync**, the administrator should ensure that the fileset is in **Active** or **NeedsResync** state. Cache can sometimes detect inconsistencies and put the fileset in **NeedsResync** state. This implies that **resync** would get run automatically during the next request to the gateway, and the inconsistencies would be resolved. If they are not resolved automatically, they have to be manually resolved by running the **resync** command. Appending to or truncating a file in SW mode will not fetch the file into cache, but will queue it to home.
Independent-writer (IW)	Allows multiple caches to write to different files. There is no locking between clusters. Each cache makes its updates to home independently. Independent writer mode supports transparent synchronization with home. That is, changes in IW cache are synchronized with home and vice versa. In case data is updated at home, all connected IW caches fetch those changes on demand based on the revalidation intervals set. Multiple active cache filesets point to one home file system or fileset and can modify the data at the same time. This mode should only be used by applications running in caches that do not require synchronization to properly update the data on the home cluster. The multiple cache sites do revalidation periodically and pull the new data from home. This mode is typically used to access different files from each IW cache site (for example, unique users at each site updating files in their home directory). While this mode allows multiple cache clusters to modify the same set of files, this should be only done by advanced users. This is because there is no locking or ordering between updates; the updates are propagated to the home in an asynchronous manner and may be delayed due to network disconnections. Therefore, conflicting updates from multiple cache sites can cause the data at the home site to be undetermined. Appending to or truncating a file in IW mode will not fetch the file into cache, but will queue it to home.

Notes:

1. SW and IW cache filesets should *not* access the same data; if they do, serious problems can occur. Only RO, LU, and IW cache filesets should access the same home dataset.
2. Resync cannot be used with IW filesets, as multiple cache filesets can update the same set of data.
3. Before converting from IW to SW mode, ensure that all the remaining IWs are converted to either RO or LU to avoid the risk of home conflicts caused by writes from other IWs. When a large number of IW caches point to the same home, the number of NFS threads at home may be tuned for better efficiency. Also, if the data updates are unique, increasing the revalidation timers for the IW caches might reduce the frequency of periodic refreshes from home and improve cache performance.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 253

File system caching and synchronization

AFM treats all file system operations as either asynchronous or synchronous.

Asynchronous operations

When an asynchronous operation is used, an application can proceed as soon as the request is queued on the gateway node. An MDS of a fileset can delay asynchronous operations for the time defined by the synchronization lag (for the time defined by the synchronization lag; this can be configured using the **mmchfileset** command).

Files and directories are refreshed from the home cluster based on a validity lag, and file system updates are synchronized to the home cluster based on the synchronization lag.

Within a single cache cluster, application nodes will experience POSIX semantics.

Data and metadata updates

All operations that modify the file system are synchronized to the home server at an interval determined by the synchronization lag. It is important to note that updates may be sent to the home cluster much sooner than the lag if proceeding operations depend on their synchronization for correctness.

AFM sends data from cache to home as root. By default, root is not limited by quotas; therefore, the fileset quotas at home could be exceeded due to writes in cache.

The asynchronous commands are: **write**, **chmod**, **chown**, **create**, **mkdir**, **remove**, **rmdir**, **rename**, **link**, **symlink**, and attribute updates.

Synchronous operations

Synchronous operations require an application request to block until the operation completes at the home cluster.

Read AFM does whole-file caching by default. In general, reading more than three blocks of the file will drive AFM to cache the full file in the background for performance. To avoid reading large files when an application might be "peeking" into the file (for example, GUI viewers that read a few bytes to detect file mime type), certain file reads at the beginning of the file are handled individually. Sequential requests are pipelined between the application and gateway nodes to improve performance. This means data can be "streamed" from the home cluster to the application at the cache cluster.

Only files that have all blocks read, or the entire file contents fetched, are marked as cached. An uncached file cannot be evicted, resynched with home, or failed over to a new home. These AFM features are discussed in subsequent sections of this guide.

File and Directory Information

As users traverse the directory tree of the AFM-enabled fileset, files and directory attribute information from the home cluster is cached in GPFS on-demand. Henceforth, commands such as **ls** or **stat** will continue to work even if the cache is subsequently disconnected from the home cluster. If file metadata information changes (via users at the home cluster or a different cache cluster), the cache is refreshed on demand to reflect new updates to existing files and creation of new files and directories. To reduce the number of operations sent to the home server, this refresh is done based on the validity lag configured. See the **afmDirLookupRefreshInterval**, **afmDirOpenRefreshInterval**, **afmFileLookupRefreshInterval**, **afmFileOpenRefreshInterval** configuration parameters documented in the *GPFS: Administration and Programming Reference*.

Update synchronization

All modifications to the cached file system will be sent back to the home cluster in the following situations:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 254

- The end of the synchronization lag.
- If a synchronous command depends on the results of one or more updates; it synchronizes all depending commands to the server prior to its execution.
- An explicit flush of all pending updates using the **mmafmctl flushPending** command.

Components of a cluster that is running AFM

Communication

Communication for caching between GPFS clusters is done by one or more gateway nodes. A node can be designated as a gateway node by using the **mmchnode** command. In addition to the gateway node designation on the cache cluster, the target path on the home server needs to be NFS-exported on any one node in the home cluster. This exported path of the home cluster would be used in the cache cluster as a target path.

Active file management uses NFSv3 or GPFS protocol for communication between clusters. AFM also keeps track of server availability and disconnects or reconnects accordingly.

Caching behavior

Active file management is designed to be efficient on a WAN by minimizing the amount of traffic sent over the WAN. To improve efficiency, there are various parameters that allow controlling caching behavior; for example, **afmFileOpenRefreshInterval** defines how often an open call checks for file changes at home.

When a file is being read into a cache file, data can be read as soon as it arrives in the cache. This means that the entire file does not have to be in cache to start reading it.

A cache is defined at the fileset level, so multiple cache filesets can share a single file system. This allows you to optimize the performance and space usage within existing storage, and it gives you flexibility in how you utilize cache relationships.

You can associate as many caches to a home as the bandwidth of the home cluster allows. The cache-to-home relationship is one-to-one, and the cache does all the work. You can think of AFM as a subscription service. The home is a data feed, and one or more caches can “subscribe” to that feed. This is what makes AFM very scalable; the GPFS cluster does not need to actively manage, as one data set, hundreds of relationships and the location on billions of files.

Gateway behavior

In a cluster with multiple gateway nodes and many AFM caches, AFM uses an improved hashing algorithm to elect the MDS for each of the caches. The improved hashing algorithm is set by default on a 4.1 cluster. On an upgraded cluster, the old hashing algorithm will be effective. To move to the improved algorithm, the **mmchconfig afmHashVersion=2** command must be run after updating to the latest release. This command needs a fileset re-link or file system remount to take effect.

When node failures occur on the MDS of a cache, AFM elects a new MDS for the cache using the hashing algorithm and runs AFM recovery on the newly elected MDS. The new MDS will thereafter take over as owner for the cache. When an old MDS returns back after node failure, AFM transfers the queues as such from the current MDS to the old MDS. A change in MDS due to a node that has changed its gateway designation using **mmchnode** will go through AFM recovery.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 255

Global namespace

Home and cache entities can be combined to create a global namespace. Any client node in the world can use the same path to connect to the data within any of the GPFS clusters that are part of the namespace. In such a global namespace, the following AFM features can improve application performance at a remote site:

- When a file is being read into a cache, the data can be read as soon as it begins to arrive in the cache.
- Multiple cache file sets can share a single file system.
- Data can be transferred between sites in parallel.

When using GPFS multi-cluster as well to mount file systems between clusters, the benefit of using AFM is performance with high latency or unreliable networks.

Figure 18 shows an example of a global namespace, implemented using AFM, with three different sites. A GPFS client node from any site sees all of the data from all of the sites. Each site has a single file system. Each site is the home for two of the subdirectories and cache filesets pointing to the data originating at the other sites. Every node in all three clusters has direct access to the global namespace.

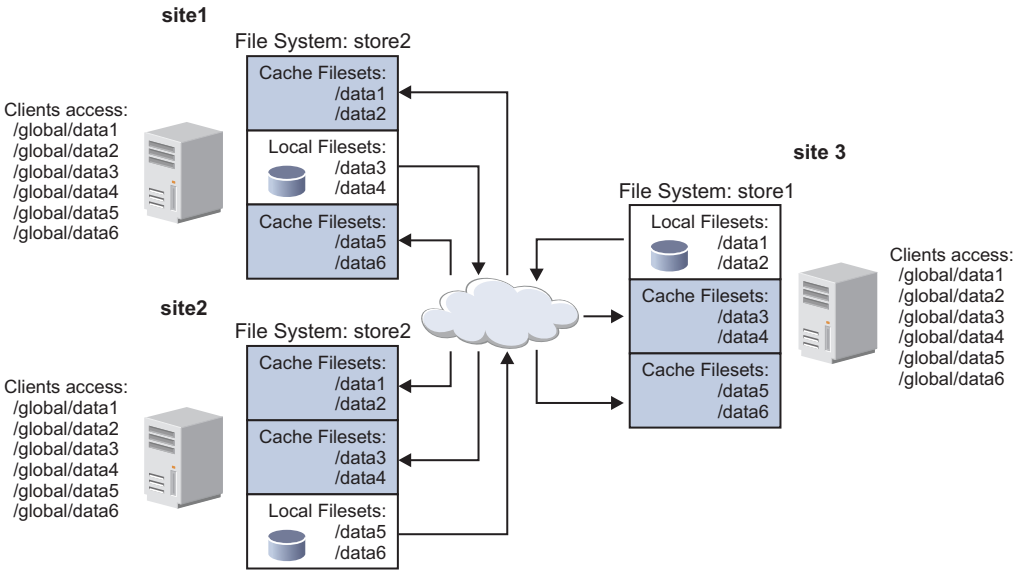


Figure 18. Global namespace implemented using AFM

Cache eviction

Cache eviction is enabled by default and is controlled by the **afmEnableAutoEviction** parameter and setting fileset quota. Eviction is enabled by default.

Cache eviction can also be triggered manually using the **mmafmctl evict** command. There might be a time lag between the time that an eviction is triggered and the time that the data is actually evicted. Tuning the soft and hard quota limits minimizes application failure caused by data being cached at a faster rate than it is being evicted. Cache eviction for exceeding the inode limit is not supported.

See “Filesets and quotas” on page 53.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 256

Disconnected operation

With the home cluster across a wide-area network, it is very likely that network connectivity will experience frequent intermittent outages and occasional long term disruptions. The reasons for these outages or disruptions can vary and can include faulty network hardware, routing or connection problems to the home cluster, NFS server failure at the home cluster, or even failure of the home cluster file system itself.

In the event that one or more gateway nodes determine that the home cluster cannot be accessed, all the caches that have a disconnected home will go into disconnected mode. The behavior in a disconnected setting is slightly different for the two classes of operations. All synchronous operations are handled locally. Application requests for objects that are not yet in the cache will return an error that the object does not exist (or an I/O error). On the other hand, all asynchronous operations can still update the cache and return successfully to the application. These requests remain queued at the gateway nodes pending reconnection. Updates will experience a synchronization lag equal to the duration of the disruption in connectivity.

| Appending to or truncating an uncached file in a writer mode (SW or IW) sends only the updated
| contents to home on reconnect. Old contents at home are lost.

| Note that since AFM caches file metadata separately from data, it is possible that one is able to see the
| attributes of a file via the ls or stat command, but not be able to fetch its contents from home, in
| disconnected mode. In the case of caches based on native GPFS protocol, unavailability of the home file
| system on the cache cluster will put the caches into unmounted state. These caches will never enter the
| disconnected state.

It is also important to note that if the cache cluster is disconnected for an extended period, the number of file system updates may exceed the buffering capacity of the gateway nodes. In this situation, all changes continue to be "logged" in stable storage. Upon reconnection, AFM reads the changes from disk and synchronizes its local updates with the home cluster.

Expiration

Read-only filesets can be configured to cause cached data to expire after the gateway nodes have been in a disconnected state for a specified amount of time. This prevents access to stale data, where staleness is defined by the amount of time that the WAN cache is out of synchronization with data at the home site. Single-writer and local-update filesets cannot be configured to expire. The administrator can manually specify that the data in the cache is either expired or not expired using the **mmafmctl expire/unexpire** command.

| Caches created using GPFS protocol will not go to expired state, even if expiration timeout is set for
| them. They will continue to remain in unmounted state as long as the home file system is unavailable.

| Cache states

An AFM cache can have a different state depending on fileset and queue states. The **mmafmctl getstate** command displays the current cache state.

Possible cache states are the following:

Active This state indicates that the cache is active and ready for operations.

Dirty This state indicates that there are pending changes in cache that are not yet played at home. This state will not hamper user function, and the user can continue normal activity on the cache.

Disconnected
This state can occur only in a cache that is created over NFS export. It occurs when the MDS

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 257

cannot connect to the NFS server at home. When parallel I/O is configured, this state shows the connectivity between the MDS and the mapped home server, irrespective of other gateway nodes. See "Parallel I/O" on page 241 for more details. To come out of this state to **Active** state, the administrator must correct the errant NFS server or servers on the home cluster.

Dropped
A cache fileset state moves to **Dropped** when there are problems in the cache during recovery or IW failback processes. The problems can include the local file system being full, no space on the cache, or policy failure during recovery. The administrator must rectify the issue and retry recovery or failback. The cache needs to be accessed to re-trigger recovery and to reissue the failback; it must be manually run using **mmafmctl**.

Dropped state is also possible under the following conditions:

- When a cache with active queue operations is forcibly unlinked. While all queued operations are being dequeued, the fileset remains in dropped state and moves to inactive state when the unlinking is complete. There is no administrative action required for this temporary dropped state.
- While AFM internally performs queue transfers from one gateway to another to handle gateway node failures. This usually gets rectified automatically on the next accessing of the cache.

Expired
This state can also occur only in a cache that is created over NFS export. It occurs when an expiration timeout is set on a fileset. All requests in the queue are dropped. To come out of this state to **Active** state, the administrator must correct the errant NFS server or servers on the home cluster.

FailbackCompleted
This state occurs when a failback has successfully completed on a cache fileset. The administrator must run **mmafmctl failback stop** to move the cache to **Active** state.

FailbackInProgress
This state occurs when a failback process has been initiated on a cache fileset and it is in progress. This will move automatically to **FailbackCompleted** once the failback process has completed.

FailoverInProgress
This state shows that the cache is in the middle of a failover process. This will move to **Active** when the failover is complete.

FlushOnly
This state indicates that operations are queued but have not started to flush. This is a temporary state and should move to **Active** when a write is initiated.

Inactive
This state is possible when a fileset has been just created and operations have not been initiated on the fileset. This should move to **Active** once operations begin.

NeedsFailback
This state can occur when a failback initiated on a cache is interrupted and is incomplete. Failback will automatically get triggered on the fileset, or the administrator can rerun failback.

NeedsResync
This state occurs when there is some accidental corruption of data of a single writer home. The **mmafmctl resync** command needs to be run on the fileset to move it to **Active** state.

Queueonly
A cache fileset is moved to **QueueOnly** when operations at the cache are queued but not yet flushed, since operations from recovery, resync, failover are in the process of getting flushed to home. This state is temporary and the user can continue normal activity.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 258

Recovery

A cache is said to be in recovery state when it is running the recovery or resync process and is in the process of queueing or flushing the updates. This state will not hamper normal user function, and the state will move to **Active** once recovery or resync is complete.

Unmounted:

Caches using NFS as transport protocol will go into **Unmounted** state if home NFS is not accessible, or if home exports are not exported properly or home export does not exist. The problem with home exports must be resolved. After 300 seconds, the cache will retry connecting with home and will move to **Active** state.

Caches using native GPFS protocol as target will move to **Unmounted** state when there are problems accessing the local mount of the remote file system. To resolve this problem, the remote file system needs to be remounted on the local cache cluster.

A cache fileset can also move to **Unmounted** state when the cache detects a change in home, such as accidental deletion of the exported independent fileset at home.

Failure and recovery

Failures can occur in AFM if either an application node or a gateway node fails. Failures are not catastrophic and will not result in the loss of data or AFM's ability to update the home cluster with local modifications. AFM stores, on disk, all necessary state to "replay" updates made to the cache at the home cluster in case the in-memory queue at the gateway node is lost due to a node failure or memory pressures.

Application node failure

Since application nodes are just regular GPFS nodes that can also access AFM filesets, failure of one of them is handled just like a regular node failure. No special handling is required for AFM.

Gateway node failure

If there were no updates made to any AFM-enabled fileset, then the failure of a gateway node is harmless and application nodes will not experience requests delays. While in recovery mode, application requests to all AFM-enabled filesets are momentarily blocked.

On the other hand, if pending updates for any AFM-enabled fileset exist, e.g., file write or create, then the failure of a gateway node puts the entire cache cluster into recovery mode.

In recovery mode, the cache examines the requests in cache that have not been sent to home and queues all these pending operations on the gateway nodes. AFM collects the pending operations by internally running a policy scan on the files in the fileset. AFM leverages the policy infrastructure in GPFS to use all the nodes mounting the file system. Therefore, the availability of all these nodes mounting the file system is required for recovery to complete. When this queue is completely played at home, according to the synchronization lag, the cache and home become synchronized in contents and recovery is said to be completed. The cache goes back to **Active** state.

Recovery is triggered only for single-writer mode or independent-writer mode. Recovery can run in parallel on multiple filesets, although only one instance can run on a fileset at one time.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 259

Steps to deal with an IW cache fileset disaster

In cases where an IW cache is down, either permanently due to disaster or temporarily due to maintenance, the applications can be moved to the home cluster. The nature of AFM is to push the data asynchronously, so the cache might have pending updates that have not been pushed to home at the time when the applications are moved. In cases where application semantics allow them to move from cache to home with pending updates still in cache, it is possible that files at home will change along with the pending data in cache.

Once the cache comes back, the first access will drive the cache to push stale data to home. This might result in a stale pending update to overwrite the latest data written at home while the cache was down. In these cases, the following steps should be followed so that latest data is preserved and old or stale data is discarded. The data staleness is determined based on file mtime and failover time (the time when applications were moved to home, given as input to failback command).

In the case of an unplanned DR, follow these administrative actions:

1. When the cache site is ready to take over again, stop applications at the home site.
2. Link the cache site back or bring the gateways up. Do not access the fileset directories.
3. Run the following command from the cache site to sync up the latest data (specifying the time when the home site became functional and including the complete time zone, in case home and cache are in different time zones):

   mmafmctl *FileSystem* **failback -j** *Fileset* **--start --failover-time** '*TimeIncludingTimezone*'

   The failback command resolves any conflicts between pending updates in cache at the time of failover with the data changes at home. Any new files created at home are fetched on next access to those respective files or directories based on the revalidation interval. In cases where an application needs to know these new files upfront, they can be fetched via **mmafmctl prefetch** before starting applications on cache. When the sync-up is in progress, the fileset state is **FailbackInProgress**.

   Note: If failback does not complete or if the fileset moves to **NeedFailback** state, rerun the failback command.

   After failback completes, the fileset state moves to **FailbackCompleted**.
4. After achieving the **FailbackCompleted** state, run this variation of the **mmafmctl failback** command to move the fileset to **Active** state:

   mmafmctl *FileSystem* **failback -j** *Fileset* **--stop**

   The cache site is ready for use. All applications can start functioning from the cache site.

   This option moves the cache from **Read-Only** state to **Active** state, where it is ready for use. Failback does not pull in uncached data from home, which needs to be done explicitly by the administrator using the **mmafmctl prefetch** utility. This can be done before running the stop option to ensure that cache has all the latest data from home before new changes are made.

As a result of an unplanned outage, some updates from the cache site might be lost. These lost updates will be handled when the cache site syncs up with the latest data updates from the home site.

The cache will identify any dirty data that did not get synced up and will do the following:

1. If the file is not modified at the home when the applications were connected to the home, the cache will push the change to the home so that there is no loss of data.
2. If the file was most recently modified at the home, the cache will discard the earlier updates from the cache. The next lookup on the cache will update the latest metadata from the home.

If the conflicted dirty data is a directory, it is moved to the `.ptrash` directory. If the conflicted dirty data is a file, it is moved to the `.pconflict` directory. The administrator will have to clean the `.pconflict` directory from time to time, as it will not be cleaned automatically. When IW is converted to other modes, the `.ptrash` and `.pconflict` directories will remain.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 260

In RO or LU mode when there is a delete conflict with a file (that is, a file is removed at home but is still existing in cache), the conflict is resolved by removing the file from the cache. In SW or IW mode, if such a file is non-dirty, it is moved to `.ptrash`, and if it is dirty, it is moved to `.pconflicts`.

Files moved back from `.ptrash` or `.pconflicts` are treated as local files to the cache fileset and are not queued to home. However, if they are copied to the cache, they will be queued to home as new files.

The fileset behaves as a read-only fileset while failback is in progress.

Creating hard links at home or in cache is not recommended when IW is used.

If failback is expected to be used, the home file system must update the ctime of the object on a RENAME operation. This is enabled via **setCtimeOnFileRename** at home.

This can be done using the **mmchconfig** command at home:

```
mmchconfig setCtimeOnFileRename=yes -i
```

Prefetching

Wide Area Network (WAN) caching fetches complete files on demand from the home file set to the remote system cache in real time during normal operation. The prefetching feature moves files into the cache in batch mode so that they are already present in the cache when they are accessed by an application. The prefetching of cache before an application is started can reduce the network delay when the application starts. Prefetching can also be used to proactively manage WAN traffic patterns by moving files over the WAN during a period of lower WAN usage in anticipation that it might otherwise be accessed during a period of higher WAN usage.

GPFS can prefetch the data using the **mmafmctl** *Device* **prefetch –j** *FilesetName* command (which specifies a list of files to prefetch). Note the following about prefetching:

- It can be run in parallel on multiple filesets (although more than one prefetching job cannot be run in parallel on a single fileset).
- If a file is in the process of getting prefetched, it will not be evicted.
- If the MDS of a cache is changed while prefetch is running, prefetch will get interrupted and will not reach completion. The next access to the fileset will automatically re-trigger the interrupted prefetch on the new MDS. For re-triggering prefetch after gateway node failures, the list file used when prefetch was initiated must exist in a path that is accessible to all gateway nodes. If parallel I/O is configured, all gateways will participate in the prefetch process.

Peer snapshots

The cache fileset peer snapshot function provides a generic infrastructure for obtaining an application-consistent point-in-time copy of data in the cache file set that can be integrated with other functions to meet specific customer requirements such as backup and recovery. The peer snapshot pushes all of the snapshot data in cache to the home fileset so that the home data is consistent with cache, then takes a snapshot of the corresponding home data. This results in a pair of peer snapshots, one each at the cache and home filesets, that refer to the same consistent copy.

Note: If cache is disconnected from the home file set when the cache snapshot is created, the cache notes that the peer snapshot on the home file set has not been created. When the home file set becomes reconnected to cache, the cache attempts to resume the creation of the missing peer snapshot at the home file set if conditions permit. The **mmpsnap** command can be used to create a peer snapshot.

The last successful snapshot is saved and can be viewed via **mmlssnapshot**. Multiple outstanding peer snapshots can be queued on the gateway node. Peer snapshots should not be deleted via the normal **mmdelsnapshot** command but rather through the **mmpsnap** command to ensure that both the cache and

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 261

home snapshots are removed. The **mmpsnap** command will work only if the home cluster has executed the **mmafmconfig enable** *ExportPath* command. **mmpsnap** can be used only in an SW cache; it cannot be used for RO, IW, or LU caches.

It is not possible to create a peer snapshot when a cache is created using GPFS protocol.

Viewing snapshots at home

All snapshots at home for RO and LU filesets can be viewed in the cache by setting the **afmShowHomeSnapshot** parameter to **yes**. This variable is not applicable for SW and IW filesets.

The home and cache can have different snapshot directories for clarity. If the **afmShowHomeSnapshot** value is changed after fileset creation, the change is reflected in cache only after the next GPFS recycle or file system remount. However, if the value is changed from **yes** to **no** at any time during the lifetime of a fileset, it will continue to show the home snapshots even after GPFS recycle or file system remount.

Failover of cache filesets

AFM filesets continue to function independent of the home, in the event of home failures. The AFM filesets serve the cache applications with cached data. When a new home replaces the failed home, the administrator should run the **mmafmctl** *Device* **failover -j** *Fileset* command to let the cache point to the new home. The home is expected to be empty when this command is run.

If failover is interrupted (for example, due to gateway node failure or quorum loss), the next access that requires access to home will restart it automatically.

Caches in SW and IW can be failed over to a new home. Failover will fill the new home with the contents from the cache. Both cached data and metadata from cache are pushed to the new home. To ensure that extended attributes get synced, the new home should have run the **mmafmconfig** command before the failover command.

When there are multiple IW caches, the administrator should choose a primary IW cache and fail this over to a new empty home. It is recommended that all the other IW caches to the old home be deleted and recreated. If another of the IW caches is failed over to the home after failing over the primary IW cache, all the stale data in that cache will overwrite the objects of the same type and name at home. For this reason, failover to a non-empty home is discouraged.

There are no implications for failover and resync when applied to a GPFS target. It is possible to change the protocol along with the target using failover. For example, a cache using an NFS target `bear110:/gpfs/gpfsA/home` can be switched to a GPFS target whose remote file system is mounted at `/gpfs/fs1`, and vice-versa as follows:

```
mmafmctl fs0 failover -j afm-mc1 --new-target gpfs:///gpfs/fs1
mmafmctl fs0 failover -j afm-mc1 --new-target nfs://bear110/gpfs/gpfsA/home
```

Note: Each AFM fileset is independently managed and has a one-to-one relationship with a target. This allows different protocol backends to coexist on separate filesets in the same file system. However, AFM does not validate the target for correctness when the fileset is created. It is the responsibility of the user to specify a valid target; using a GPFS target that belongs to the same file system as the AFM fileset may lead to undesirable consequences.

Partial file caching

Partial file caching fetches only blocks that are accessed, thereby utilizing network and local disk space more efficiently. This is useful if an application does not need to read the whole file. Partial file caching is enabled on a GPFS block boundary.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 262

For sparse files, the percentage for prefetching is calculated as the ratio of the size of data blocks allocated in cache and the total size of data blocks at home. Holes at the home file are not considered in the calculation.

Partial file caching is controlled by the **afmPrefetchThreshold** configuration attribute. For details, see the **mmchfileset** command description in *GPFS: Administration and Programming Reference*.

Prefetch of partially cached files pulls the complete file to the cache. Failover of partially cached files pushes only the cached data blocks to the new home. Uncached blocks are filled with null bytes.

Any writes queued on a partially cached file will pull in the whole file, even if a prefetch threshold is set on the file.

Parallel I/O

The gateway acting as the MDS of a cache fileset is the channel for communication with the home cluster. To help the MDS exchange data with the home cluster for large-size I/Os, a cache cluster can be configured to leverage all nodes in the cluster configured as gateways.

For this function, multiple NFS servers are required at the home cluster. In a cache cluster that uses NFS as transport protocol, each gateway node in the cache cluster can be mapped to a specific NFS server at home. This mapping aids I/O load distribution.

One or more gateway nodes can be mapped to the same NFS server, but one gateway node cannot be mapped to more than one NFS server. Mapping defined using **mmafmconfig** needs a GPFS recycle to take effect on existing filesets. If there is no mapping specified or if a matching failed, parallelism will not be effective.

On an AFM cluster using native GPFS protocol for defining an AFM target, gateway nodes can be mapped to any other node in the same cache cluster. In the absence of a mapping definition, all gateway nodes will be used for the I/O.

The following shows an example mapping for NFS target, assuming cache gateway servers `hs22n18`, `hs22n19`, `hs22n20`, and `hs22n21`, and home NFS servers `js22n01` and `js22n02`:

```
# mmafmconfig add js22n01 --export-map js22n01/hs22n18,js22n02/hs22n19
mmafmconfig: Command successfully completed
mmafmconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmafmconfig add js22n02 --export-map js22n02/hs22n20,js22n01/hs22n21
mmafmconfig: Command successfully completed
mmafmconfig: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

The **mmafmconfig** command can be used to display, delete, or update mappings. New changes in the active mapping between gateway nodes and home NFS servers will take effect only after fileset re-link or file system remount. Gateway designation can only be removed from a node if the node is not participating in an active mapping.

```
# mmafmconfig show
Map name:             js22n01
Export server map:    192.168.200.12/hs22n19.gpfs.net,192.168.200.11/hs22n18.gpfs.net

Map name:             js22n02
Export server map:    192.168.200.11/hs22n20.gpfs.net,192.168.200.12/hs22n21.gpfs.net
```

Parallel reads and writes need to be configured separately. Parallel reads or writes will be effective for files whose sizes are larger than those specified by the respective parallel threshold. The threshold can be defined using **afmParallelWriteThreshold** and **afmParallelReadThreshold** parameters. This will hold

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 263

good for reads and writes on all types of files except reads on sparse files and reads on files with partial file caching enabled, which will be served only by the MDS without splitting.

The MDS of a cache fileset communicates with the each of the participating gateway nodes based on their availability. A single I/O request will be split into multiple chunks. These chunks will be sent across all gateway nodes in parallel. The size of each chunk is configurable using the **afmParallelWriteChunkSize** and **afmParallelReadChunkSize** parameters.

The gateway nodes will communicate with their respective NFS servers and get a reply. If more than one gateway node is mapped to the same NFS server at home, only one among the gateway nodes will be chosen to perform a read task. However a write task will be split among all the gateway nodes. This is not applicable to caches created using native GPFS protocol, which will use the mapping specified or all gateway nodes wherever no mapping is specified.

The MDS will process the replies from all gateway nodes. The MDS handles all I/O failures from the participating gateway nodes and coordinates all the activities until the IO is completed

Configuration parameters for parallel reads and writes are discussed in detail in "Tuning active file management home communications" on page 249.

Disabling AFM

An AFM fileset can be converted to a normal independent fileset. An independent fileset cannot be converted to an AFM fileset. In scenarios where all data from home is destroyed or not available, the AFM fileset can be converted to an independent fileset by using the **mmchfileset -p afmTarget=disable** option after unlinking the fileset. Thereafter, the fileset will behave like a regular GPFS independent fileset and no requests will be queued to home.

Disabling targets for AFM filesets is intended only for RO or LU filesets, not for writer filesets. This feature is useful in NFS migration use cases, where AFM can be disabled after migration. Migration is discussed in detailed in NFS migration section.

Hierarchical storage management and AFM

Hierarchical storage management (HSM) is a data storage technique that automatically moves data between high-cost and low-cost storage media. AFM does not support migration of files to an IBM Tivoli Storage Manager HSM server, and it is not recommended to migrate using HSM on AFM filesets. This can be achieved by using the **EXCLUDE** rule on the link directory of AFM filesets. If HSM is used on AFM filesets, it will result in recalling data from the AFM home server and not from the HSM server.

mmbackup and AFM

When the **mmbackup** command is run on a file system that has AFM filesets, only cached data from the AFM filesets get backed up; **mmbackup** will not back up uncached data from the AFM filesets.

AFM-based NFS migration

NFS Migration is a process of migrating data from any legacy storage appliance to a GPFS cluster, via NFS protocol. The old storage appliance can be disconnected once the migration is complete. If an IP switchover is subsequently executed, the migration will be seamless and users or applications will not be affected. AFM NFS Migration can be very useful in the event of upgrading hardware or buying a new system where the data from old hardware needs to be moved to new hardware.

AFM provides an option to do planned migration via a prefetch command or dynamically (by pulling data on demand based on file access), mitigating total downtime required for migration. In the case of dynamic migration, the data is moved slowly based on access over a period of time (ranging from days

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 264

to weeks or months, based on network speed and other factors). This is called progressive migration. Once the data is completely moved, the source of migration can be completely removed. Also, planned and dynamic migration can be mixed based on application or administration requirements.

The key goals of data migration include minimizing down time for applications and moving data along with attributes. Data migration aims at potentially consolidating data from multiple legacy systems to a more powerful system, keeping configuration data intact.

In order to use migration, the target or new hardware should be running GPFS 4.1 or later. The data source should be an NFS v3 export and can be either a GPFS or a non-GPFS source. If the source is running a version of GPFS earlier than GPFS 3.4, it is equivalent to a non-GPFS data source.

On a GPFS data source, AFM moves all GPFS extended attributes and ACLs, and file sparseness is maintained. It does not migrate file system–specific parameters like quotas, snapshots, file system level tuning parameters, policies, fileset definitions, encryption keys, dmapi parameters, and so on. For hard links to be maintained, prefetching has to be done with the **--ns** option. See the **mmafmctl prefetch** command description in *GPFS: Administration and Programming Reference*.

AFM migrates data as root, bypassing permission checks. Preallocated files at home source are not maintained; that is, only actual data blocks are migrated based on file size. UID namespace between source and target must be maintained.

On a non-GPFS data source, POSIX permissions or ACLs are pulled. AFM will not migrate NFS V4/CIFS ACLs, non-posix file system attributes, and sparseness.

Before starting migration, the administrator should gather the requirements to choose the right method to migrate. If home is running GPFS 3.4 or later, run **mmafmconfig --enable** at home before starting migration. **afmAutoEviction** must be disabled at fileset creation time. The data source NFS export should be configured as read-only, as AFM will not push any data to the source system. During creation of an AFM fileset, it is advisable to preallocate inodes equal to or greater than the inodes at source, as that many files are known to exist. The maximum inodes should also be set accordingly.

Following are the migration steps for a generic example where the source is a legacy storage device and the target is an IBM GPFS cluster with AFM:

- On the home cluster (Legacy NAS Storage), the NFS shares whose data must be migrated are identified.
- All identified shares are exported via NFS.
- At the cache cluster (GPFS AFM), an AFM fileset is created for each NFS share exported from the home cluster.
- The mode of the filesets is LU or RO, depending upon the type of access required.
- All applications can be moved from pointing to the data source to the cache cluster (GPFS cluster).
- While users access applications, AFM prefetch tasks can be initiated in parallel on the GPFS cluster to pull data from the source.
- After all data is brought into the GPFS AFM cluster, AFM can be disabled. This will complete the data migration process.

Migration options

The following migration options are available:

Incremental migration

To perform an incremental migration, follow these steps:

1. Create an AFM RO fileset with the AFM target pointing to NFS v3 export at data source.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 265

2. Make a file list using a simple LIST RULE via policy if the home is GPFS, or using **find** or **ls -lR** or any similar tool, and feed this file list to **mmafmctl --prefetch --ns**. This will populate the directory tree in the RO fileset.
3. The administrator can migrate either some files selectively or all files using **mmafmctl --prefetch --filelist**.
4. Once prefetch completes and invokes the callback, the migration of given files is complete.
5. Repeat steps 2 through 5, making incremental file lists using policy or other tools to feed into **--prefetch**, getting the target as close to the source as possible.
6. At some point where source and target are close to each other, take a downtime and stop applications at home. Run one last incremental step.
7. Once migration is complete, run a policy **MISC_ATTRIBUTES NOT like %u%** to list any uncached files. If the list is empty, all migration is complete and files have their data completely in cache.
8. Disable AFM via **mmchfileset** to use as a regular fileset, and move applications to the target or new system. The old system is ready to be decommissioned.

In this option, applications are down while doing the last incremental data migration.

Progressive migration

To perform a progressive migration, follow these steps:

1. Create an AFM LU fileset with the AFM target pointing to NFS v3 export at data source.
2. Make a file list using a simple LIST RULE via policy if the home is GPFS, or using **find** or **ls -lR** or a similar tool and feed this file list to **mmafmctl --prefetch --ns**. This will populate the directory tree in LU fileset.
3. Stop applications, make a diff list between step 2 and step 3, and run **prefetch --ns** again to get the namespace in sync.
4. Start applications in cache. The downtime is only between step 2 & 3, which can be minimized if run incrementally also. With a single namespace, this is much quicker than a full data prefetch.
5. The actual file data will be migrated on demand when needed by an application or when the administrator can drive a background prefetch via **mmafmctl --prefetch cmd**.
6. At some point, when data has been completely migrated (known by running a policy **MISC_ATTRIBUTES NOT like %u%**), disable AFM via **mmchfileset**.

In this option, applications are down only during the last incremental namespace migration (to deal with failure cases). For example: During migration, where some portions of a directory tree are not yet migrated, if the source gets disconnected and applications see that a file X does not exist and creates it. Once the network is back, file X exists at home also, resulting in a conflict. In this case, data from source file X is never migrated. In some instances this might be acceptable, but in others, the application would not have created one if file existed at home and might have determined its course of action based on its contents. Also in the case of disconnection during migration, application access to a uncached file (that is, a file that has not been migrated yet) will get an E_IO error.

Progressive migration with no downtime

To perform a progressive migration with no downtime, follow these steps:

1. Create an AFM LU fileset with the AFM target pointing to NFS v3 export at data source.
2. Move applications from the data source to the AFM cache fileset or move critical file data via **mmafmctl --prefetch** and then move applications.
3. Data will be migrated on demand, and the administrator can migrate critical data that does not need latency of access from the source via **mmafmctl --prefetch**.

The downtime required is only for moving applications to new cluster. In this case, applications will experience latency to pull data from source on demand. The key here is to make sure that

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 266

data source or AFM target is not disconnected during migration to avoid the application getting errors when accessing uncached files and to avoid any namespace conflicts due to disconnection.

The UIDs should be maintained across the source and target cluster. That is, if a file is owned by UID 10, then the same UID will own it in cache; so its very important that both systems have same UID-to-user mapping.

At the time of pull, the mtime of files are maintained from source to target.

Using HSM to migrate

In case data is managed via HSM on the source, migration will drive data recall from tape onto source and then to the target system, or similarly, if the source disk space is small and HSM can be enabled at the target.

Only files that have been completely migrated are eligible for HSM migration to tape. Partially migrated or files that have not been migrated to tape will have the Windows offline bit set, making them ineligible for HSM migration to tape.

Once the file is migrated to tape at the target, the next access to the migrated file will trigger dmapi recall from the HSM data source and will present it to applications.

Notes:

1. Callbacks can be added to indicate when a prefetch task is completed (thereby indicating when migration is completed).
2. 2. After migration, AFM filesets can be converted to SW or IW filesets with a new target, or to regular GPFS filesets with caching disabled.
3. 3. Parallel I/O can be used for migration.

Restrictions in active file management

In AFM, certain file attributes are maintained locally in the cache and separately in the home site; they are not transmitted to the other side.

The file attributes that are maintained this way include the following:

- control attributes
- direct I/O
- replication factors
- fileset quotas
- storage pool flags

The same is also true of certain special file types, including the following:

- FIFO
- socket
- block device
- character device

Snapshots on either side are also not played on the other side. To create peer snapshots in single-writer mode, you must use the **mmpsnap** command. Use the *snapdir* variable for specifying the snapshot location because the reserved **.snapshots** directory is the same at both the cache and home systems.

The following are not supported by AFM:

- hard links

  The hard links at the home site are not copied as hard links to cache. If a hard link is created in cache, that relationship is also created at the home site.
- clones

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 267

The **mmclone** command is disabled from cache. Clones created at home are not displayed as clones in the cache system; instead, the cache system displays the clones as separate files.

- append-only or immutability flags

  When a file is **appendonly** or **immutable** at home, the cache reflects these attributes. Setting a file as **immutable** from cache sets the attribute but does not queue the request to home. Setting a file as **appendonly** is not allowed from cache.

The clone, snapshot, hard link and rename restrictions also apply to data that is prefetched from the home site to the cache site.

The following are other AFM restrictions:

1. When AFM is recovering from a failure, all WAN caching application requests are blocked momentarily.
2. It is the responsibility of the customer to ensure replication between Lightweight Directory Access Protocol (LDAP) servers used by the home and cache system.
3. Data transfer rates are not guaranteed.
4. The cache eviction function requires that quotas are enabled at the cache level.
5. AFM does not support file locking across systems. If an application locks a cached file at the cache system only the cached file is locked; the corresponding file at the home system is not locked by this request.
6. AFM does not support NFS delegation.
7. If a file is renamed at the home site, the file in cache is deleted and created again in cache. This results in the file being assigned a different inode number at the cache site.
8. There is no option to encrypt or compress AFM data.
9. Conversion of AFM filesets is supported on unmounted file systems. To convert an AFM fileset, the fileset must be unlinked, converted, and linked back.
10. Failover is only possible for SW or IW filesets. RO and LU filesets cannot be failed over.
11. For AFM caches using native GPFS protocol as target, AFM puts a hold on the remote vfsmount point. Hence it is not possible to unmount the remote file system until the relevant AFM cache is unlinked or local file system is unmounted.
12. Parallel I/O is supported only in Linux-only clusters, and all nodes should have the same architecture (x86 or ppc).
13. If home is a mix of architectures (x86 and ppc), then parallel I/O will work only for the set of nodes that belong to any one architecture, depending on which architecture serves the I/O first.
14. From Linux 2.6 onwards, all home NFS exports should mention the FSID value. See the topic about "Linux export considerations" in *GPFS: Administration and Programming Reference*.
15. Recovery on a cache fileset that has at any point contained millions of inodes will take a long time.
16. Creating an AFM association, using GPFS protocol, to the same local file system is not allowed.
17. Fileset snapshot restore is not supported for AFM filesets.

Home cluster errors

When a synchronous command experiences an error (such as reading a cached file while disconnected, or permission errors with the home cluster), it immediately reports the error back to the user.

If an asynchronous update (such as a create, write, or attribute change) fails due to any reason (for example, **EPERM**, **EROFS**, **EINVAL**, or configuration errors), an error message is written to the log, and the operation is left on the queue and put into a special state. All further operations that depend on the failed operation will not execute. These errors are logged in the `/var/adm/ras/mmfs.log.latest` file on the MDS of a fileset or a fileset gateway.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 268

When this happens, you can look at the error in the log file and try to fix the problem. Once the problem is fixed, you can execute the **mmafmctl resumeRequeued** command on the gateway node with the failed message. This will put the operation back in a normal state and try to execute it to the home cluster. If another error occurs, or the problem is still not fixed, the whole process starts over again.

If the process of fixing the error in the cache involves a GPFS restart or gateway reboot, then the queue might get dropped. This would put the cache out of sync with home, as discussed in "Failure and recovery" on page 237. The gateway would go into recovery mode when it comes up again.

If an abnormal error such as **E_STALE** is received from home for a write operation in IW mode, the cache will cause the write to be dropped, with no recourse. An AFM error will be logged in this situation, and the file will be pulled in from home on the next revalidation. No corrective action needs to be taken by the administrator for these errors.

Administrative actions

This section describes the administrative actions associated with AFM.

System setup and requirements

Enabling a cache on an existing file system may involve some additional steps and possibly tuning.

Defining gateway nodes

Gateway nodes are needed on the cache sides to be able to communicate to home. An NFS client should be available on the gateway nodes, and they need to be able to mount the NFS-exported path of the home cluster.

When a gateway node is added to the cluster, it is initially inactive and will not be used to communicate with the home cluster. Once the GPFS daemon on the gateway node is restarted (or simply started if it was previously down), it is activated and is available for AFM functionality.

Use the **mmchnode** command to define a node as a gateway node. For example, to define `node1` as a gateway node, run the following command:

```
mmchnode --gateway –N node1
```

Adding or removing nodes

In essence, an application node is no different from a node in a standard GPFS file system. Therefore, the addition or removal of an application node is the same as the addition or removal of a standard GPFS node.

Setting up home and cache clusters

This topic explains how to set up home and cache clusters and enable cache cleaning.

Setting up the home cluster

When configuring a file system on the home cluster for external access, you must complete the following steps on all nodes that will be used to export the file system:

1. Start the file system.
2. Set NFS export information. Ensure that the following items are true:
   - The directory within the file system to be cached is exported and is configured with the correct access permissions.
   - Gateway nodes at cache site have access to the exported directory (can be mounted using NFS).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 269

Note: The **no_root_squash** and **sync** NFS export arguments are critical.

For single-writer filesets, ensure that the file system is exported with **rw**. For read-only filesets, the file system can be exported with either **rw** or **ro**.

3. Start NFS services. This will start all necessary NFS programs and export the file system.

   Note: If the file system goes down for any reason, the exported file system must be re-exported.

4. Enable the exported path at home suitable for AFM, using the following command at the home cluster:

   mmafmconfig enable *ExportPath*

   See the **mmafmconfig** command description in *GPFS: Administration and Programming Reference*.

This is the GPFS startup sequence at home to prepare file system `gpfsA` (Linux):

```
# Start daemons
mmstartup -a

# Sample /etc/exports file contents:
/gpfs/gpfsA/ *(fsid=12345,rw,sync,insecure,no_root_squash,nohide,no_wdelay)

# Start nfs services
/etc/init.d/nfs start
```

Note: If both NFS and GPFS start automatically at bootup time, it is very important to ensure that GPFS starts before NFS. This is because NFS can only export GPFS if it is loaded. If NFS starts before GPFS, **exportfs -r** needs to be executed.

The home side of the relationship is defined at the NFS share level. The home contains all of the data available from the NFS mount point. It is a best practice to create one or more filesets for export on the home cluster, define NFS mount points at each fileset junction, and export each fileset individually.

Setting up a cache cluster

Before setting up a cache cluster, complete the following preparation steps:

1. Determine which nodes of the GPFS cluster will be application nodes and which will be gateway nodes.
2. Create an AFM fileset and link it.
3. Configure **maxfilestocache** to allow queueing of a large number of pending requests at gateway nodes.

Follow these steps to set up the cache cluster:

1. Prior to starting the GPFS daemon on the chosen gateway nodes, specify the gateway nodes:

   mmchnode --gateway -N *Node1*,*Node2*,...

2. Ensure that GPFS has been started:

   mmstartup -a

3. Mount the file system:

   mmmount *Device*

4. Create an AFM-enabled fileset. There is no special file system configuration required to be a cache. The cache relationship is defined when you create a fileset by specifying the **mmcrfileset -p** parameter and specifying the required attribute **afmTarget** and the optional **afmMode** attribute and **--inode-space** parameter.

   mmcrfileset *Device Fileset* **-p afmTarget=***Home-Exported-Path* **--inode-space=new**
   -p afmMode=single-writer | read-only | local-updates | independent-writer

5. Link the fileset to the file system:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 270

mmlinkfileset *device fileset* **-J /gpfs/gpfsA/fs1**

Once the fileset is linked you are ready to start caching data.

Enabling cache cleaning

Cache cleaning is a feature where file data blocks are deallocated to make room for new files when fileset usage exceeds the fileset soft quota. This process is of deallocating blocks is called eviction. Note that the deallocation of file blocks is not done if file data is dirty and is not in sync with home. This feature is enabled only when fileset soft quotas are enabled.

To enable cache cleaning, enable a fileset soft quota for the cache file set by specifying the **-Q** option on the **mmcrfs** or **mmchfs** commands. Cleaning starts when fileset usage reaches the soft quota limit.

Dealing with requeued messages

When there is an error in executing a message, AFM re-queues the message. AFM tries to execute the message at a later point in time, and this continues in a loop until a message gets executed. In some cases, the user or administrator can correct the reason for the error and then force the message to be executed immediately, without waiting for AFM to execute in the next cycle. The **mmafmctl resumeRequeued** command can be used to execute the requeued messages in the queue.

Tuning active file management home communications

Note: For most recent tuning recommendations, see the GPFS FAQ in the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) or GPFS FAQ in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfsclustersfaq.html).

AFM communication is done using the NFSv3 protocol; for peak performance it is a good idea to tune the gateway NFS servers that are hosting home exports and the gateway servers supporting cache filesets.

Most of these tuning parameters require at least the AFM client to be restarted. It is always safest to make sure that the NFS server is not mounted. You can achieve this by unlinking the AFM filesets or stopping GPFS on the gateway node.

There are significant differences between tuning for 1GigE and 10GigE networks. For 10GbE everything needs to be scaled up, but not necessarily by a factor of 10. Many of these settings do not survive reboots so they need to be set each time the server is started. The TCP buffer tuning is required for 1GigE links with rtt > 0 and for all 10GigE links (possibly even in a LAN).

Tuning on gateway nodes of the cache cluster (NFS Client)

The **sunrpc.tcp_slot_table_entries** or **/proc/sys/sunrpc/tcp_slot_table_entries** parameter sets the maximum number of (TCP) RPC requests that can be in flight. Leave as the default for 1GigE with no round-trip time. You may want to increase it beyond 16 if the round-trip time is large. For 10GigE, make sure this value is at least 48; possibly even higher depending on the round-trip time.

Tuning required on both the NFS client and the NFS server

You must set TCP values that are appropriate for the delay (buffer size = bandwidth * RTT).

For example, if your ping time is 50 ms, and the end-to-end network consists of all 100BT Ethernet and OC3 (155 Mbps), the TCP buffers should be the following:

```
0.05 sec * 10 MB/sec = 500 KB
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 271

If you are connected via a T1 line (1 Mbps) or less, the default buffers are fine; but faster networks usually benefit from buffer tuning.

The following parameters can also be used for tuning (note that a 12194304 buffer size is provided here as an example value for a 1GigE link with a delay of 120ms). To set these values, place the following settings in a file, then load it with **sysctl -p** *filename*.

The following are example values; initial testing may be required to determine the best values for a particular system:

```
net.ipv4.tcp_rmem = 12194304 12194304 12194304
net.ipv4.tcp_wmem = 12194304 12194304 12194304
net.ipv4.tcp_mem = 12194304 12194304 12194304
net.core.rmem_max = 12194304
net.core.wmem_max =  12194304
net.core.rmem_default = 12194304
net.core.wmem_default = 12194304
net.core.optmem_max = 12194304
net.core.netdev_max_backlog = 30000
net.ipv4.tcp_no_metrics_save = 1
net.ipv4.tcp_timestamps = 0
net.ipv4.tcp_sack = 1
```

Note: For TCP tuning, the **sysctl** value changes will not take effect until a new TCP connection is created (which occurs at NFS mount time). Therefore, for TCP changes, it is critical that the AFM fileset (and thus the NFS client) is unmounted and GPFS is shut down.

With RHEL 6.1 and later, both the NFS client and server perform TCP autotuning. This means that it will automatically increase the size of the TCP buffer within the specified limits (via **sysctl**). If either the client or server TCP limits are too small, the TCP buffer will grow as needed for various round-trip time between GPFS clusters. With versions earlier than RHEL6.1, NFS is limited in its ability to tune the TCP connection. Therefore, do not use a version earlier than RHEL6.1 in the cache cluster.

Since a GPFS cluster may be handling local and remote NFS clients, one strategy is to set the GPFS server values at the value for the largest expected round-trip time of any NFS client. This will ensure that the GPFS server can handle clients in various locations. Then on the NFS clients, set the TCP buffer values that are appropriate for the SONAS cluster that they are accessing.

For AFM, the gateway node is both an NFS server (for standard NFS clients if they exist) and an NFS client (for communication with the home cluster). Ensure that the TCP values are set appropriately, as values that are either too big or too small can negatively impact performance.

If performance continues to be an issue, try increasing the buffer value by *at most* 50%. Any further increase in size will have a negative effect on performance.

NFS server tuning on the home cluster

/proc/fs/nfsd/max_block_size
: Set this to 1MB for improved performance.

/proc/fs/nfsd/threads
: Set to a minimum value of 32, but possibly greater than 128 depending on the throughput capacity and round-trip time between the cache and home clusters. You might need to experiment to determine the correct value.

nfsPrefetchStrategy
: Since AFM uses NFS, ensuring that this is set on the home GPFS cluster is critical. Set it to at least 5 to 10.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 272

Tuning active file management

The following table lists AFM configuration parameters with their default values and the commands used to set them initially or change them.

Table 40. Active file management configuration parameters and default values

AFM configuration parameter	Valid values	Default value	Notes
afmAsyncDelay	1 through 2147483647	15	This is an optional attribute that is set by default when the fileset is created.
afmDirLookupRefreshInterval	0 through 2147483647	60	This is an optional attribute that is set by default when the fileset is created.
afmDirOpenRefreshInterval	0 through 2147483647	60	This is an optional attribute that is set by default when the fileset is created.
afmDisconnectTimeout	0 through 2147483647, **disable**	60	This is an optional attribute that is set by default when the fileset is created. Only **mmchconfig** can be used to change this attribute.
afmExpirationTimeout	0 through 2147483647, **disable**	300	This is an optional attribute that is set by default when the fileset is created.
afmFileLookupRefreshInterval	0 through 2147483647	30	This is an optional attribute that is set by default when the fileset is created.
afmFileOpenRefreshInterval	0 through 2147483647	30	This is an optional attribute that is set by default when the fileset is created.
afmMode	**read-only** | **ro**, **local-updates** | **lu**, **single-writer** | **sw**, **independent-writer** | **iw**	**read-only** | **ro**	This is a required attribute for the **mmcrfileset** command.

Active file management settings

Once the NFS values have been set, you can mount and access the AFM filesets. The first time the fileset is accessed the AFM NFS client mounts the home server or servers. To see these mounts on a gateway node, enter the following command:

```
cat /proc/mounts
```

The system displays the mount point and the mount options. If the **wsize** and **rsize** values are not 1MB, you can adjust the parameters and remount to get the correct values.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 273

Tuning on the cache cluster

When the GPFS parameter **seqDiscardThreshold** has been set, it affects AFM as follows:

- If I/O requests are from a node other than the gateway node, there is no effect.
- If the read request is made on the gateway node for an uncached file, a higher **seqDiscardThreshold** value results in higher performance. This is because it allows the gateway to cache more data, which means that when the data is returned to the application, there is a greater chance that it comes out of the cache.

AFM performance tuning parameters at cache

The following parameters can be used for performance tuning at cache

afmHardMemThreshold
Sets the maximum amount of memory that AFM can utilize on each gateway node to record changes to the file system. Once this limit is reached, the fileset goes into "dropped" state. Exceeding the limit can occur if the cache cluster is disconnected for an extended period of time or if the connection with the home cluster has low bandwidth. The fileset will be brought back to "active" state once the fileset goes into connected mode.

afmNumFlushThreads
Defines the number of threads used on each gateway to synchronize updates to the home cluster. The default value is 4, which is sufficient for most installations. The current maximum value is 32, which is too high for most installations; setting this parameter to such an extreme value should be avoided.

afmNumReadThreads
This tunable defines how many read threads can be used on each participating gateway node during parallel read. The default value of this parameter is 4; that is, four reader threads will be active on every gateway node for each big write operation qualifying for splitting per the parallel read threshold value. The valid range of values is 1 to 64.

afmNumWriteThreads
This tunable defines how many write threads can be used on each participating gateway node during parallel write. The default value of this parameter is 4; that is, four writer threads will be active on every gateway node for each big write operation qualifying for splitting per the parallel write threshold value. Valid values can range from 1 to 64.

afmParallelReadChunkSize
Defines the minimum chunk size of the read that needs to be distributed among the gateway nodes during parallel reads. Values are interpreted in terms of bytes. The default value of this parameter is 128 MB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the **mmchconfig** command. It can be set at fileset level using **mmcrfileset** or **mmchfileset** commands.

afmParallelReadThreshold
Defines the threshold beyond which parallel reads become effective. Reads are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MB. The default value is 1024 MB. The valid range of values is 0 to 2147483647. It can be changed cluster wide with the **mmchconfig** command. It can be set at fileset level using **mmcrfileset** or **mmchfileset** commands.

afmParallelWriteChunkSize
Defines the minimum chunk size of the write that needs to be distributed among the gateway nodes during parallel writes. Values are interpreted in terms of bytes. The default value of this parameter is 128 MB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the **mmchconfig** command. It can be set at fileset level using **mmcrfileset** or **mmchfileset** commands.

afmParallelWriteThreshold
Defines the threshold beyond which parallel writes become effective. Writes are split into chunks when file size exceeds this threshold value. Values are interpreted in terms of MB. The default value

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 274

of this parameter is 1024 MB, and the valid range of values is 0 to 2147483647. It can be changed cluster wide with the **mmchconfig** command. It can be set at fileset level using **mmcrfileset** or **mmchfileset** commands.

afmReadSparseThreshold
This parameter specifies the file size beyond which sparseness of a file in cache is maintained. File sparseness in cache is detected and maintained when the size of allocated blocks in the cache for a file exceeds **afmReadSparseThreshold**. If the size of a file is less than the threshold, sparseness is not maintained.

The default value of this parameter is 128M. It can be changed with the **mmchconfig** command.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 275

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 276

Chapter 14. GPFS File Placement Optimizer

GPFS File Placement Optimizer (FPO) is a set of features that allow GPFS to operate efficiently in a system based on a shared nothing architecture. It is particularly useful for "big data" applications that process massive amounts of data.

FPO makes use of the following entities and policies:

Chunks

A chunk is a logical grouping of blocks that allows the grouping to behave like one large block, particularly useful for applications that need high sequential bandwidth. Chunks are specified by a block group factor that dictates how many file system blocks are laid out sequentially on disk to behave like a large block. Chunks can be specified on a storage pool basis or at a file level.

On the file level, the block group factor can be specified by the **--block-group-factor** argument of the **mmchattr** command. The range of the block group factor is from 1 to 1024; the default value is 1. You can also specify this through the **blockGroupFactor** argument in a storage pool stanza (as input to the **mmadddisk** or **mmcrfs** command).

The effective large block size is a function of both block group factor and block size. For example, setting block size to 1 MB and block group factor to 128 will lead to an effective large block size of 128 MB.
See the following command descriptions in *GPFS: Administration and Programming Reference*:

- **mmadddisk**
- **mmchattr**
- **mmcrfs**

Extended failure groups

A failure group is defined as a set of disks that share a common point of failure that could cause them all to become simultaneously unavailable. Traditionally, GPFS failure groups have been identified by simple integers. In an FPO-enabled environment, a failure group may be specified as not just a single number, but as a vector of up to three comma-separated numbers that convey topology information that GPFS will exploit when making data placement decisions.

In general, a topology vector is a way for the user to specify which disks are closer together and which are farther away. In practice, the three elements of the failure group topology vector may represent the rack number of a disk, a position within the rack, and a node number. For example, the topology vector 2,1,0 identifies rack 2, bottom half, first node.

When considering two disks for striping or replica placement purposes, it is important to understand the following:

- Disks that differ in the first of the three numbers are farthest apart (as they are in different racks).
- Disks that have the same first number but differ in the second number are closer (as they are in the same rack, but in different halves).
- Disks that differ only in the third number reside in different nodes in the same half of the same rack.
- Only disks that have all three numbers in common reside in the same node.

The data block placement decisions are also affected by the level of replication and the value of the **writeAffinityDepth** parameter. For example, when using replication 3, GPFS may place two replicas far apart (different racks) to minimize chances of losing both. On the other hand, the third replica may be placed close to one of the others (same rack, but different half), to reduce network traffic between racks when writing the three replicas.

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 277

To specify the topology vector that identifies a failure group, you use the **failureGroup=***FailureGroup* attribute in an NSD stanza (as input to the **mmadddisk** or **mmcrfs** command).

See the following command descriptions in *GPFS: Administration and Programming Reference*:

- **mmadddisk**
- **mmcrfs**

Write affinity depth

Write affinity depth is a policy that indicates that the node writing data directs the write to its own node for the first copy, and to the disks in other nodes for the second and third copy (if specified). The policy allows the application to determine the layout of a file in the cluster so as to optimize for typical access patterns. The write affinity is specified by a depth that indicates how many copies are localized (as opposed to wide striped) and can be specified at the storage pool or file level.

To specify write affinity depth, you use the **writeAffinityDepth** attribute in a storage pool stanza (as input to the **mmadddisk** or **mmcrfs** command) or the **--write-affinity-depth** argument of the **mmchattr** command.

A write affinity depth of 0 indicates that each replica is to be striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group. By default, the unit of striping is a block; however, if the block group factor is specified in order to exploit chunks, the unit of striping is a chunk.

A write affinity depth of 1 indicates that the first copy is written to the writer node. The second copy is written to a different rack. The third copy is written to the same rack as the second copy, but on a different half (which can be composed of several nodes).

A write affinity depth of 2 indicates that the first copy is written to the writer node. The second copy is written to the same rack as the first copy, but on a different half (which can be composed of several nodes). The target node is determined by a hash value on the fileset ID of the file, or it is chosen randomly if the file does not belong to any fileset. The third copy is striped across the disks in a cyclical fashion with the restriction that no two disks are in the same failure group.

This behavior can be altered on an individual file basis by utilizing the **--write-affinity-failure-group** option of the **mmchattr** command.

Note: Write affinity depth of 2 is designed to assign (write) all the files in a fileset to the same second-replica node. However, this behavior depends on node numbers in the cluster; after a node is added to or deleted from a cluster, a different node is selected as the second replica for files in a fileset.

See the description of storage pool stanzas that follows. Also, see the following command descriptions in *GPFS: Administration and Programming Reference*:

- **mmadddisk**
- **mmchattr**
- **mmcrfs**

Write affinity failure group

Write affinity failure group is a policy that indicates the range of nodes (in a shared nothing architecture) where replicas of blocks in a particular file are to be written. The policy allows the application to determine the layout of a file in the cluster so as to optimize for typical access patterns.

You specify the write affinity failure group through the **write-affinity-failure-group** *WafgValueString* attribute of the **mmchattr** command. Failure group topology vector ranges specify the nodes, and the specification is repeated for each replica of the blocks in a file.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 278

For example, the attribute `1,1,1:2;2,1,1:2;2,0,3:4` indicates that the first replica is on rack 1, rack location 1, nodes 1 or 2; the second replica is on rack 2, rack location 1, nodes 1 or 2; and the third replica is on rack 2, rack location 0, nodes 3 or 4. The default policy is a null specification, which indicates that each replica is to be wide striped over all the disks in a cyclical fashion such that no two replicas are in the same failure group.
When data in an FPO pool is backed up in a TSM server and then restored, the original placement map will be broken unless you set the write affinity failure group for each file before backup.

Note: To change the failure group in a write-affinity–enabled storage pool, you must use the **mmdeldisk** and **mmadddisk** commands; you cannot use **mmchdisk** to change it directly.
See the following command descriptions in *GPFS: Administration and Programming Reference*:

- **mmadddisk**
- **mmchattr**
- **mmdeldisk**

Enabling the FPO features

To efficiently support write affinity and the rest of the FPO features, GPFS internally requires the creation of special allocation map formats. When you create a storage pool that is to contain files that make use of FPO features, you must specify **allowWriteAffinity=yes** in the storage pool stanza.

To enable the policy to read replicas from local disks, you must also issue the following command:

```
mmchconfig readReplicaPolicy=local
```

See the description of storage pool stanzas that follows. Also, see the following command descriptions in *GPFS: Administration and Programming Reference*:

- **mmadddisk**
- **mmchconfig**
- **mmcrfs**

Storage pool stanzas

Storage pool stanzas are used to specify the type of layout map and write affinity depth, and to enable write affinity, for each storage pool.

Storage pool stanzas have the following format:

```
%pool:
  pool=StoragePoolName
  blockSize=BlockSize
  usage={dataOnly | metadataOnly | dataAndMetadata}
  layoutMap={scatter | cluster}
  allowWriteAffinity={yes | no}
  writeAffinityDepth={0 | 1 | 2}
  blockGroupFactor=BlockGroupFactor
```

See the following command descriptions in *GPFS: Administration and Programming Reference*:

- **mmadddisk**
- **mmcrfs**

Recovery from disk failure

A typical shared nothing cluster is built with nodes that have direct-attached disks. Disks are not shared between nodes as in a regular GPFS cluster, so if the node is inaccessible, its disks are also inaccessible. GPFS provides means for automatic recovery from these and similar common disk failure situations.

The following command sets up and activates the disk recovery features:

```
mmchconfig restripeOnDiskFailure=yes -i
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 279

Whether a file system is a subject of a recovery attempt is determined by the max replication values for the file system. If the **mmlsfs -M** or **-R** value is greater than one, then the recovery code will be executed. The recovery actions are asynchronous and GPFS will continue its processing while the recovery attempts take place. The results from the recovery actions and any errors that are encountered will be recorded in the GPFS logs.

Two additional parameters are available for fine tuning the recovery process:

```
mmchconfig metadataDiskWaitTimeForRecovery=seconds
mmchconfig dataDiskWaitTimeForRecovery=seconds
```

The default value for **metadataDiskWaitTimeForRecovery** is 1800 seconds. The default value for **dataDiskWaitTimeForRecovery** is 3600 seconds.
See the following command description in *GPFS: Administration and Programming Reference*:

- **mmchconfig**

Distributing data across a cluster

There are a few ways to distribute data uniformly across a cluster:

- Import the data through a diskless GPFS client; this ensures that the data is distributed evenly across all failure groups and all nodes within a failure group.
- Use a write affinity depth of 0 across the cluster; however, the data distribution tends to favor the failure group of the ingest node.
- Make every GPFS node an ingest node and deliver data equally across all ingest nodes; however, this strategy is expensive in terms of implementation.

Ideally, all the failure groups should have an equal number of disks with roughly equal capacity. If one failure group is much smaller than the rest, it is likely to fill up faster than the others, and this will complicate rebalancing actions.

After the initial ingesting of data, the cluster may be unbalanced. In such a situation, use the **mmrestripefs** command with the **-b** option to rebalance the data.

FPO pool file placement and AFM

For AFM home or cache, an FPO pool file written on the local side will be placed according to the write affinity depth and write affinity failure group definitions of the local side. When a file is synced from home to cache, it follows the same FPO placement rule as when written from the gateway node in the cache cluster. When a file is synced from cache to home, it follows the same FPO data placement rule as when written from the NFS server in the home cluster.

To retain the same file placement at both home and cache, ensure that each has the same cluster configuration, and set the write affinity failure group for each file.

Restrictions

The following restrictions apply in an FPO environment:

- Storage pool properties can be set only when the pool is created and cannot be changed later.
- All disks in an FPO pool must be assigned an explicit failure group.
- All disks in an FPO pool must have exactly one NSD server associated with them.
- All disks in an FPO pool that share an NSD server must belong to the same failure group.
- When replacing a disk in an FPO pool, the old and new disks must have the same NSD server.
- Disks must be removed from the file system before NSD servers can be changed.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 280

There may be additional limitations and restrictions. For the latest support information, see the GPFS FAQ in the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) or GPFS FAQ in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfsclustersfaq.html).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 281

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 282

Chapter 15. Encryption

GPFS provides support for file encryption that ensures both secure storage and secure deletion of data. GPFS manages encryption through the use of encryption keys and encryption policies.

Note: GPFS encryption is only available with GPFS Advanced Edition. The file system must be at the latest version for GPFS 4.1. Encryption is supported in multicluster environments (provided that the remote nodes have their own `/var/mmfs/etc/RKM.conf` files and access to the remote key managers; see "Encryption keys") and FPO environments, but encryption cannot be applied to AFM-managed filesets.

Secure storage uses encryption to make data unreadable to anyone who does not possess the necessary encryption keys. The data is encrypted while "at rest" (on disk) and is decrypted on the way to the reader. Only data, not metadata, is encrypted.

GPFS encryption can protect against attacks targeting the disks (for example, theft or acquisition of improperly discarded disks) as well as attacks performed by unprivileged users of a GPFS node in a multi-tenant cluster (that is, a cluster that stores data belonging to multiple administrative entities called tenants). However, it cannot protect against deliberate malicious acts by a cluster administrator.

Secure data deletion leverages encryption and key management to guarantee erasure of files beyond the physical and logical limitations of normal deletion operations. If data is encrypted, and the master key (or keys) required to decrypt it have been deleted from the key server, that data is effectively no longer retrievable. See "Encryption keys."

Important: Encryption should not be viewed as a substitute for using file permissions to control user access.

Encryption keys

GPFS uses the following types of encryption keys:

master encryption key (MEK)

An MEK is used to encrypt file encryption keys.

MEKs are stored in remote key management (RKM) servers and are cached by GPFS components. GPFS receives information about the RKM servers in a separate `/var/mmfs/etc/RKM.conf` configuration file. Encryption rules present in the encryption policy define which MEKs should be used, and the `/var/mmfs/etc/RKM.conf` file provides a means of accessing those keys. The `/var/mmfs/etc/RKM.conf` also specifies how to access RKMs containing MEKs used to encrypt files created under previous encryption policies.

An MEK is identified with a unique *Keyname* that combines the name of the key and the RKM server on which it resides. See "Encryption policy rules" on page 262 for *Keyname* format.

file encryption key (FEK)

An FEK is used to encrypt sectors of an individual file. It is a unique key that is randomly generated when the file is created. For protection, it is encrypted (or "wrapped") with one or more MEKs and stored in the `gpfs.Encryption` extended attribute of the file.

A wrapped FEK cannot be decoded without access to the MEK (or MEKs) used to wrap it. Therefore, a wrapped FEK is useless to an attacker and does not require any special handling at object deletion time. If necessary, an FEK can be rewrapped using a new set of MEKs to allow for operations like MEK expiration and rotation, compromised key removal, and data expiration.

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 283

Note: If an encryption policy specifies that an FEK be wrapped multiple times, only one of the wrapped-FEK instances needs to be unwrapped for the file to be accessible.

Encryption policies

GPFS uses encryption policies to manage aspects of how file encryption is to be implemented, including the following:

- which files are to be encrypted
- which algorithm is to be used for the encryption
- which MEK (or MEKs) are to be used to wrap the FEK of a file

Encryption policies are configured using the **mmchpolicy** command and are applied at file creation time. When a file is created, encryption rules are traversed in order until one of the following occurs:

- The last rule is reached.
- The maximum number of **SET ENCRYPTION** rules that can be matched (eight) is reached.
- An **ENCRYPTION EXCLUDE** rule is matched.

If the file matches at least one **SET ENCRYPTION** rule, an FEK is generated and used to encrypt its contents. The FEK is wrapped once for each policy it matches, resulting in one or more versions of the encrypted FEK being stored in the `gpfs.Encryption` extended attribute of the file.

Notes:

1. When an encryption policy is changed, the changes apply only to the encryption of subsequently created files.
2. Encryption policies are defined on a per–file system basis by a system administrator. Once the encryption policies are put in place, they may result in files in different filesets or with different names being encrypted differently.

Encryption policy rules

GPFS provides the following rules with which you can specify encryption policies:

ENCRYPTION IS

This rule is used to specify how a file is to be encrypted and how the FEK is to be wrapped.

The syntax of the **ENCRYPTION IS** rule is:

```
RULE 'RuleName' ENCRYPTION 'EncryptionSpecificationName' IS
    ALGO 'EncParamString'
    COMBINE 'CombineParamString'
    WRAP 'WrapParamString'
    KEYS('Keyname'[, 'Keyname', ... ])
```

where:

ALGO *EncParamString*

specifies the encryption parameter string, which defines the following:

- encryption algorithm
- key length
- mode of operation
- key derivation function

The following encryption parameter strings are valid:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 284

Table 41. Valid EncParamString values

Value	Description
AES:128:XTS:FEK:HMACSHA512	Encrypt the file with AES in XTS mode. The FEK is 128 bits long and is preprocessed using HMAC with SHA-512.
AES:256:XTS:FEK:HMACSHA512	Encrypt the file with AES in XTS mode. The FEK is 256 bits long and is preprocessed using HMAC with SHA-512.
AES:128:CBC:FEK:HMACSHA512	Encrypt the file with AES in CBC mode. The FEK is 128 bits long and is preprocessed using HMAC with SHA-512.
AES:192:CBC:FEK:HMACSHA512	Encrypt the file with AES in CBC mode. The FEK is 192 bits long and is preprocessed using HMAC with SHA-512.
AES:256:CBC:FEK:HMACSHA512	Encrypt the file with AES in CBC mode. The FEK is 256 bits long and is preprocessed using HMAC with SHA-512.

COMBINE *CombineParamString*
specifies a string that defines the mode to be used to combine MEKs specified by the **KEY** statement.

The following combine parameter string values are valid:

Table 42. Valid combine parameter string values

Value	Description
XORHMACSHA512	Combine MEKs with a round of XOR followed by a round of HMAC with SHA-512.
XOR	Combine MEKs with a round of XOR.

WRAP *WrapParamString*
specifies a string that defines the encryption algorithm and the wrapping mode to be used to wrap the FEK.

The following wrapping parameter string values are valid:

Table 43. Valid wrapping parameter string values

Value	Description
AES:KWRAP	Use AES key wrap to wrap the FEK.
AES:ECB	Use AES in ECB mode to wrap the FEK.
AES:CBCIV	Use AES in CBC-IV mode to wrap the FEK.

KEYS ('*Keyname***'[, '***Keyname***', ...])**
specifies one or more keys to be applied. Each *Keyname* is a unique identifier that combines the name of the key and the RKM server on which it resides. The format for *Keyname* is:

KeyId:*RkmId*

where

KeyId
represents an internal identifier that uniquely identifies the key inside the RKM. Valid

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 285

characters for *KeyId* are the following: 'A' through 'Z'; 'a' through 'z'; '0' through '9'; and '-' (hyphen). The minimum length of *KeyId* is one character; the maximum length is 42 characters.

RkmId
corresponds to the `RKM.conf` entry of the RKM that manages the key. Valid characters for *RkmId* are the following: 'A' through 'Z'; 'a' through 'z'; '0' through '9'; and '_' (underscore). The minimum length of *RkmId* is one character; the maximum length is 21 characters.

Notes:

1. The maximum number of keys you can specify with the **ENCRYPTION IS** rule is eight.
2. The number of keys that can be used to encrypt a single file is permanently limited by the inode size of the file system.
3. You cannot specify the same key more than once in a given **ENCRYPTION IS** rule. Also, do not specify keys with identical values in an **ENCRYPTION IS** rule. Specifying the same key or identically-valued keys could result in a security breach for your data.

SET ENCRYPTION

The **SET ENCRYPTION** rule is similar to the **SET POOL** rule. If more than one such rule is present, all **SET ENCRYPTION** rules are considered and the FEK is wrapped once for each of the rules that apply (up to the maximum of eight). As mentioned in "Encryption keys" on page 261, if an FEK is wrapped multiple times, only one of the wrapped-FEK instances needs to be unwrapped for the file to be accessed.

If no **SET ENCRYPTION** rule is applicable at create time, the file is not encrypted.
The syntax of the **SET ENCRYPTION** rule is:

```
RULE 'RuleName' SET ENCRYPTION 'EncryptionSpecificationName'[, 'EncryptionSpecificationName',...]
    [FOR FILESET ('FilesetName'[,'FilesetName']...)]
    [WHERE SqlExpression]
```

where:

EncryptionSpecificationName
is the name of a specification defined by an **ENCRYPTION IS** rule.

To stop traversing policy rules at a certain point and encrypt using only those rules that have matched up to that point, use the **SET ENCRYPTION EXCLUDE** rule:

```
RULE ['RuleName']  SET ENCRYPTION EXCLUDE
     [FOR FILESET ('FilesetName'[,'FilesetName']...)]
     [WHERE SqlExpression]
```

Default encryption parameters

To simplify policy management, GPFS accepts the special default value `'DEFAULTNISTSP800131A'` as the **ALGO** parameter string.

For example, this policy statement:

```
RULE 'somerule' ENCRYPTION 'somename' IS
    ALGO 'DEFAULTNISTSP800131A'
    KEYS('KEY-2f1f7700-de74-4e55-a9be-bee49c5b3af8:RKMKMIP3')
```

corresponds to this:

```
RULE 'somerule' ENCRYPTION 'somename' IS
    ALGO 'AES:256:CBC:FEK:HMACSHA512'
    COMBINE 'XORHMACSHA512'
    WRAP 'AES:CBCIV'
    KEYS('KEY-2f1f7700-de74-4e55-a9be-bee49c5b3af8:RKMKMIP3')
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 286

Note: When this special **ALGO** default value is set as the **ALGO** *EncParamString*, neither **COMBINE** nor **WRAP** should be specified.

Example of an encryption policy

This is an example of an encryption policy:

```
RULE 'myEncRule1' ENCRYPTION 'E1' IS
          ALGO 'DEFAULTNISTSP800131A'
          KEYS('1:RKM_1', '2:RKM_2')

RULE 'myEncRule2' ENCRYPTION 'E2' IS
          ALGO 'AES:256:XTS:FEK:HMACSHA512'
          COMBINE 'XOR'
          WRAP 'AES:KWRAP'
          KEYS('3:RKM_1')

RULE 'myEncRule3' ENCRYPTION 'E3' IS
          ALGO 'AES:128:CBC:FEK:HMACSHA512'
          COMBINE 'XORHMACSHA512'
          WRAP 'AES:CBCIV'
          KEYS('4:RKM_2')

RULE 'Do not encrypt files with extension enc4'
          SET ENCRYPTION EXCLUDE
          FOR FILESET('fs1')
          WHERE NAME LIKE '%.enc4'

RULE 'Encrypt files with extension enc1 with rule E1'
          SET ENCRYPTION 'E1'
          FOR FILESET('fs1')
          WHERE NAME LIKE '%.enc1'

RULE 'Encrypt files with extension enc2 with rule E2'
          SET ENCRYPTION 'E2'
          FOR FILESET('fs1')
          WHERE NAME LIKE '%.enc2'

RULE 'Encrypt files with extension enc* with rule E3'
          SET ENCRYPTION 'E3'
          FOR FILESET('fs1')
          WHERE NAME LIKE '%.enc%'
```

Note:

In this example encryption policy:

- All files in fileset `fs1` are treated as follows:
  - If the extension is equal to `enc4`, the file is not encrypted. This happens because the `ENCRYPTION EXCLUDE` rule is matched first, stopping the traversal of the remaining rules before any additional matches can be made.
  - If the extension is equal to `enc1`, the file is encrypted with a 256-bit FEK, using AES in XTS mode; the FEK is preprocessed with HMAC with SHA-512, and the FEK is then wrapped twice:
    - once with AES key wrap, with keys `1:RKM_1` and `2:RKM_2` combined via one round of XOR followed by one round of HMAC with SHA-512
    - once with AES in CBC-IV mode using key `4:RKM_2`

    This happens because both rules `E1` and `E3` apply, since extension `enc1` matches both `%.enc1` and `%.enc%`. Note that the encryption algorithms specified by rule `E3`, which grant a stronger security than those of rule `E1`, are chosen and applied.
  - If the extension is equal to `enc2`, the file is encrypted with a 256-bit FEK, using AES in XTS mode; the FEK is preprocessed with HMAC with SHA-512; and the FEK is then wrapped twice:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 287

- once with AES key wrap using key 3:RKM_1
- once with AES in CBC-IV mode using key 4:RKM_2

This happens because both rules E2 and E3 apply, since extension enc2 matches both %.enc2 and %.enc%.

– If the extension is equal to enc3, the file is encrypted with a 128-bit FEK, using AES in CBC mode; the FEK is preprocessed with HMAC with SHA-512; and the FEK is then wrapped once with AES in CBC-IV mode using key 4:RKM_2.

This happens because only rule E3 applies, since extension enc3 only matches %.enc%.

- A GPFS node with access to both keys 1:RKM_1 and 2:RKM_2 or to key 4:RKM_2 can access a file with extension enc1.
- A GPFS node with access to key 3:RKM_1 or to key 4:RKM_2 can access a file with extension enc2.
- A GPFS node with access to key 4:RKM_2 can access a file with extension enc3.
- No key is required to access a file with extension enc4.
- A file with extension enc1 is securely deleted when either key 1:RKM_1 or 2:RKM_2, and key 4:RKM_2 are destroyed in their respective RKMs (and their cached copies have been flushed).
- A file with extension enc2 is securely deleted when key 3:RKM_1 and key 4:RKM_2 are destroyed in their respective RKMs (and their cached copies have been flushed).
- A file with extension enc3 is securely deleted when key 4:RKM_2 is destroyed in its respective RKM (and its cached copies have been flushed).
- Once created, a file may not be encrypted with more MEKs, only with different MEKs using the **REWRAP** rule.

Rewrapping policies

Rewrapping policies are used to change the way a set of FEKs is encrypted; that is, to change the set of MEKs that wrap the FEKs of those files. Rewrapping applies only to files that are already encrypted, and the rewrapping operation acts only on the gpfs.Encryption EA of the files. Rewrapping is done by using the **mmapplypolicy** command to apply a set of policy rules containing one or more CHANGE ENCRYPTION KEYS rules. These rules have the form:

```
RULE 'ruleName' CHANGE ENCRYPTION KEYS FROM 'Keyname_1' to 'Keyname_2'
[FROM POOL 'poolName']
  [FOR FILESET(...)]
  [SHOW(...)]
  [WHERE ... ]
```

where:

- The FOR FILESET and WHERE clauses narrow down the set of affected files.
- *Keyname_1* is the unique identifier of the MEK to be replaced. (See "Encryption policy rules" on page 262 for *Keyname* format.)
- *Keyname_2* is the unique identifier of the new MEK, which will replace the old MEK identified by *Keyname_1*.

Both *Keyname_1* and *Keyname_2* are listed, and only the files that currently use *Keyname_1* will have their FEKs rewrapped with *Keyname_2*. Files that do not currently use *Keyname_1* are not affected by the operation.

Notes:

1. Only the *first* matching **CHANGE ENCRYPTION KEYS** rule will be applied to each file. The rule will rewrap each wrapped version of the FEK that was encrypted with the MEK in the **CHANGE ENCRYPTION KEYS** rule.
2. The same MEK cannot be used more than once in a particular wrapping of the FEK.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 288

Encryption setup requirements

Setup requirements for encryption are as follows:

GPFS software
GPFS 4.1 Advanced Edition is required for encryption.

GPFS node setup
The following are required:

- operating system image with a running version of **mmfsd** deployed on a set of nodes
- a GPFS cluster configured and up and running
- file system at the latest version level for GPFS 4.1, with the following:
  - fast external attributes enabled. This is the default for newly-created 4.1 file systems. However, if the file system was migrated from an earlier level, it may be necessary to issue the **mmmigratefs** *FsName* **--fastea** command.
  - 4KB inodes (recommended minimum size). The 4KB inode size is recommended to accommodate the `gpfs.Encryption` extended attribute that is assigned to each encrypted file at file creation time. This extended attribute will contain one or more wrapped FEKs, so it can potentially grow quite large. (See "Encryption policies" on page 262.)
- `gpfs.gskit` and `gpfs.crypto` packages installed

RKM Setup
The following are required:

- RKM server. The only server currently supported is IBM Security Key Lifecycle Manager (ISKLM) v2.5.0.1 or later.

  Note: ISKLM has a complete implementation of the Key Management Interoperability Protocol (KMIP) standard of the Organization for the Advancement of Structured Information Standards (OASIS). GPFS nodes use the KMIP protocol to retrieve keys from ISKLM servers.
- GPFS nodes with direct network access to the RKM server

RKM Backends
Nodes in the cluster that will be used to perform data access (reading and writing files) or participate in administration tasks such as snapshot deletion, disk repair, or file system check must perform encryption. These nodes must therefore be able to contact one or more RKM backends to transparently access keys required by the encryption policies.

The configuration of each RKM backend must be described in a `/var/mmfs/etc/RKM.conf` file on each such node. The `/var/mmfs/etc/RKM.conf` files *do not* have to be identical; by controlling the contents of this file, the cluster administrator can control which client nodes have access to which keys. For example, the same RKM server could be given two different names in `/var/mmfs/etc/RKM.conf` stanzas, allowing the administrator to partition a set of MEKs hosted on a single RKM server into separate subsets of MEKs, which might belong to subsets of the nodes of the cluster.

/var/mmfs/etc/RKM.conf files

`/var/mmfs/etc/RKM.conf` files consist of a series of stanzas, each of which uses the following syntax:

```
# RKM entry
RkmId {
   type = ISKLM                          # The RKM is an ISKLM
   kmipServerUri = tls://host:port       # TLS connection to host on port
   keyStore = /PathToKeyStoreFile        # The path to the PKCS#12 file containing server certificate
                                         #  and client public/private keypair
   passphrase = Password                 # Passphrase protecting the key store file
   clientCertLabel = LabelName           # Label of the client key pair to be used among those in the
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 289

```
                                        #  key store file
    tenantName = NameOfTenant           # Name of tenant, set in IBM Security Key Lifecycle Manager set-up
    [connectionTimeout = ConnectionTimeout]   # Connection timeout, in seconds (default 60 seconds)
    [connectionAttempts = ConnectionAttempts] # Number of connection attempts (default 3)
    [retrySleep = RetrySleepUsec]       # Retry sleep time, in microsecsonds
                                        #  (default 100000 = 0.1 seconds)
}
```

Notes:

1. Each entry in the /var/mmfs/etc/RKM.conf file starts with an *RkmId* string that identifies the RKM. (This is the same string that will later be combined with unique *KeyId* identifiers to specify individual keys in encryption policy rules. See "Encryption policy rules" on page 262.)
2. The following limits apply:
   - 0 < *ConnectionAttempts* <= 10
   - 0 < *RetrySleepUsec* <= 10000000 (in microseconds)
   - 0 < *ConnectionTimeout* <= 120 (in seconds)
3. The file size of the /var/mmfs/etc/RKM.conf file cannot exceed 1MiB.
4. There is no limit on the number of /var/mmfs/etc/RKM.conf entries (as long as the file size does not exceed the file size limit of 1 MiB).
5. The /var/mmfs/etc/RKM.conf file and the key store file are essential for security. It is best practice to store them as node-local files.

Identifying multiple RKM backends in a high-availability configuration

The ISKLM supports automated replication across multiple nodes for high-availability deployments. To identify multiple RKM backends in a high-availability configuration, specify any of the following optional parameters:

```
rkmname3 {
...
   kmipServerUri2 = tls://host:port   # TLS connection to clone number 1 to host on port
   kmipServerUri3 = tls://host:port   # TLS connection to clone number 2 to host on port
   kmipServerUri4 = tls://host:port   # TLS connection to clone number 3 to host on port
   kmipServerUri5 = tls://host:port   # TLS connection to clone number 4 to host on port
   kmipServerUri6 = tls://host:port   # TLS connection to clone number 5 to host on port
...
}
```

If at least one backup is configured, whenever key retrieval from the master fails, GPFS will attempt to fetch the key from each backup in turn until it is able to retrieve the desired MEK. Note that the addition of the URIs for the clone servers is the only required change within GPFS; all other configuration parameters (certificates, keys, node and tenant info) need not change, as they are also part of the set of information that is replicated. The administrator is responsible for creating and maintaining any backups.

Additionally, setting up ISKLM key server clones can help gain some performance advantage by distributing MEK retrieval requests across the different clones in a round-robin fashion. To achieve this, the administrator must specify different orderings of the server endpoints on different GPFS nodes in the /var/mmfs/etc/RKM.conf file.

For example, if there are two cloned ISKLM servers available (such as tls://keysrv.ibm.com:5696 and tls://keysrv_backup.ibm.com:5696), half of the nodes in the cluster can have the following content in /var/mmfs/etc/RKM.conf:

```
...
   kmipServerUri  = tls://keysrv.ibm.com:5696
   kmipServerUri2 = tls://keysrv_backup.ibm.com:5696
...
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 290

The other half can use the following:

```
...
   kmipServerUri  = tls://keysrv_backup.ibm.com:5696
   kmipServerUri2 = tls://keysrv.ibm.com:5696
...
```

Establishing an encryption-enabled environment

Establishing an encryption-ready environment requires the sequence of activities summarized here. This summary covers a basic setup with a single encrypted fileset.

Considerations on FIPS compliance

To retain compliance to FIPS 140-2, do the following:

- Ensure that the `fips` configuration parameter is turned to **on** on the ISKLM *before* generating server certificates and MEKs. (See the ISKLM installation guide for more information.)
- Ensure that GPFS is running with the **FIPS1402mode** configuration variable set to **yes** *before* creating the key store file.

Initial setup by administrator

During initial setup, perform the following steps:

1. **Install IBM Security Key Lifecycle Manager (ISKLM) version 2.5.0.1 or later.**

   Refer to the ISKLM installation guide.

   From the main page of the ISKLM web GUI, select **Configuration** and then **Key Serving Parameters** from the selector on the right. Then make sure that the check box next to **Keep pending client device communication certificates** is selected.
2. **Configure an SSL/KMIP server certificate on ISKLM.**

   (Skip this step if you have already configured the SSL/KMIP server certificate).

   a. Click **Configuration** in the menu bar at the top, and select SSL/KMIP from the menu on the left.

   b. To create a new self-signed certificate, click the **Create self-signed certificate** radio button.

   c. Enter the appropriate details for the certificate to be created, and click **OK**.
3. **Export the SSL/KMIP server certificate from ISKLM.**

   This step is required to obtain a trusted copy of the server certificate.

   a. Identify the certificate label for the SSL/KMIP server certificate currently in use.

   b. Click **Advanced Configuration** from the menu bar at the top, and select **Server Certificates** from the dropdown menu.

   c. Select the certificate identified as being "In Use," click **Modify**, and make note of the certificate label.

   d. From an administrator console on the ISKLM server, change the directory to the `bin` directory in `WAS_HOME`:

      - Windows systems:

        `drive:\Program Files (x86)\IBM\WebSphere\AppServer\bin`

      - Other systems such as Linux:

        `/opt/IBM/WebSphere/AppServer/bin`

   e. Type the following:

      - On Windows systems:

        `wsadmin -username SKLMAdmin -password mypwd -lang jython`

      - On other systems such as AIX or Linux:

        `./wsadmin.sh -username SKLMAdmin -password mypwd -lang jython`

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 291

f. Identify the correct UUID of the certificate:

1) First issue the following **print** command, replacing *YOUR_LABEL* with the certificate label obtained in the previous step:

```
print AdminTask.tklmCertList('[-alias YOUR_LABEL]')
```

The system will respond with output similar to the following:

```
CTGKM0001I Command succeeded.
uuid = CERTIFICATE-7005029a-831d-405f-af30-4bf0177909de
alias = server
key store name = defaultKeyStore
key state = ACTIVE
issuer name = CN=server
subject name = CN=server
creation date = 13/03/2014 16:27:13 Eastern Daylight Time
expiration date = 09/03/2015 07:12:30 Eastern Daylight Time
serial number = 1394363550
```

2) Reissue the **print** command, this time specifying the UUID identified during the previous step:

```
print AdminTask.tklmCertExport('[-uuid
CERTIFICATE-7005029a-831d-405f-af30-4bf0177909de -format base64 -fileName
/root/srvcert]')
```

The certificate will be exported in the file specified after `-fileName` (in this example, `/root/srvcert`).

g. Copy the file into a temporary directory on the GPFS node you wish to configure for encryption (for example, to `/tmp/srvcert`).

4. **Create a new GPFS device group.**

a. From the main page of the ISKLM web GUI, select **Advanced Configuration** and then **Device Group** from the menu bar at the top.

b. On the next page, click the **Create** button.

c. In the **Create Device Group** window, select the **GPFS** device family and enter an appropriate name (for example, "GPFS_TENANT1"); make note of this name, as it will be required later on.

d. Once the device group has been created, the system will prompt you to add devices and keys. However, the next steps in this procedure will generate keys explicitly, so now just click **Close** to return to the main page.

e. On the main page, select the GPFS tenant device group, click **Go to...**, and select **Manage keys and devices** from the dropdown menu.

f. On the next screen, click **Add** and select **Key** from the dropdown menu.

The system prompts you to specify the following:

- the number of keys to be created
- the three-letter prefix to be added to their name

The name is the ISKLM internal name and is not used for GPFS encryption.

Make note of the key UUID (in the example, `KEY-326a1906-be46-4983-a63e-29f005fb3a15`), as it will be required later on.

g. Enable the option **Hold new certificate requests pending my approval** by selecting it from the dropdown menu in the lower part of the page.

5. **Configure the RKM backend on GPFS.**

a. Create a key store containing the server certificate and a new key pair and certificate that will be used by the client.

1) To do so, from an administrator console on the GPFS node you wish to be configured with encryption, create the subdirectory `/var/mmfs/etc/RKMcerts`.

2) Issue the following command, replacing *cname_test* with a string identifying the GPFS node or the tenant performing encryption, *label_test* with a string that identifies the client certificate in the key store, and *a_password* with a password protecting the secret key material in the key store file:

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 292

```
mmauth gencert --cname cname_test --label a_label --cert /tmp/srvcert --out
/var/mmfs/etc/RKMcerts/ISKLM.p12 --pwd a_password
```

Upon success, the new key store is created (for example, in `/var/mmfs/etc/RKMcerts/ISKLM.p12`).

3) Open an editor to create the `/var/mmfs/etc/RKM.conf` file, as follows:

```
ISKLM_srv {
type = ISKLM
kmipServerUri = tls://raclette.zurich.ibm.com:5696
keyStore = /var/mmfs/etc/RKMcerts/ISKLM.p12
passphrase = a_password
clientCertLabel = a_label
tenantName = GPFS_TENANT1
}
```

Choose an appropriate name for the RKM backend stanza (`ISKLM_srv` in the example) and ensure that the fields are filled in correctly:

type Must be set to ISKLM.

kmipServerUri
Contains the DNS name or IP address of the ISKLM server and the KMIP SSL port (you can find the latter in the main page of the ISKLM web GUI; 5696 is the default).

keyStore
Contains the path to the key store file created in the previous step.

passphrase and clientCertLabel
Contain the password and label specified in the command line upon creation of the key store.

tenantName
Contains the name of the device group created in ISKLM.

6. **Set up an encryption policy.**

a. Create a policy, instructing GPFS to encrypt all files in the file system with an FEK and wrap the FEK with the MEK created previously; for example:

```
RULE 'p1' SET POOL 'system' # one placement rule is required at all times

RULE 'Encrypt all files in filesystem with rule E1'
      SET ENCRYPTION 'E1'
      WHERE NAME LIKE '%'

RULE 'simpleEncRule' ENCRYPTION 'E1' IS
      ALGO 'DEFAULTNISTSP800131A'
      KEYS('KEY-326a1906-be46-4983-a63e-29f005fb3a15:ISKLM_srv')
```

b. Create the *Keyname* string in the policy statement as follows:

KeyID
The key ID copied from the ISKLM web GUI.

RkmID
The name of the RKM backend stanza in the `/var/mmfs/etc/RKM.conf` file.

c. After installing the rule with **mmchpolicy**, you need to import the client certificate into ISKLM. To do that, attempt the creation of an encrypted file to trigger a KMIP request, for example:

```
[root@baden ~]# touch /gpfs0/test
touch: cannot touch `/gpfs0/test': Permission denied
[root@baden ~]# tail -n 2 /var/adm/ras/mmfs.log.latest
Thu Mar 20 14:00:55.029 2014: [E] Unable to open encrypted file: inode 46088,
Fileset fs1, File System gpfs0.
Thu Mar 20 14:00:55.030 2014: [E] Error: key
'KEY-326a1906-be46-4983-a63e-29f005fb3a15:ISKLM_srv' could not be fetched (RKM
reported error -1004).
```


Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 293

The request will initially fail because ISKLM does not yet trust the client certificate; however, the certificate will be placed in a list of pending certificates.

d. From the main page of the ISKLM web GUI, select **Pending client device communication certificates**.

e. Identify the client certificate and click **View**.

f. Carefully check that the certificate being imported matches the one created in the previous step, then click **Accept and Trust**

g. On the resulting screen, give a name to the certificate, then click **Accept and Trust**.

h. Attempt again the file creation that previously failed to verify that the setup is complete:

```
[root@baden ~]# touch /gpfs0/test
[root@baden ~]# mmlsattr -n gpfs.Encryption /gpfs0/test
file name: /gpfs0/test
gpfs.Encryption: "EAGC????f?????????????? ??????w?^??>???????????? ?L4??
_-???V}f???X????,?G?<sH??0?)??M?????)?KEY-326a1906-be46-4983-a63e-29f005fb3a15?
isklmsrv?)?KEY-6aaa3451-6a0c-4f2e-9f30-d443ff2ac7db?RKMKMIP3?"
EncPar 'AES:256:XTS:FEK:HMACSHA512'
        type: wrapped FEK WrpPar 'AES:KWRAP' CmbPar 'XORHMACSHA512'
                KEY-326a1906-be46-4983-a63e-29f005fb3a15:isklmsrv
```

All files created from this point on will be encrypted with an FEK, which in turn will be wrapped with the chosen MEK.

Security note: The contents of `/var/mmfs/etc/RKM.conf` and of the key store file (`/var/mmfs/etc/RKMcerts/ISKLM.p12` in the example) are extremely security-sensitive. Make sure file permissions are appropriately set, for example:

```
-rw-------. 1 root root 2446 Mar 20 12:15 /var/mmfs/etc/RKM.conf
drw-------. 2 root root 4096 Mar 20 13:47 /var/mmfs/etc/RKMcerts
-rw-------. 1 root root 3988 Mar 20 13:47 /var/mmfs/etc/RKMcerts/ISKLM.p12
```

Also, make sure the passphrase is not leaked through other means (for example, the shell history). Finally, note that GPFS neither manages nor replicates these files, so make sure you take appropriate precautions to ensure that they are not lost or corrupted.

Enabling encryption on other nodes

If you want to replicate the same encryption configuration on another node, you can simply replicate the content of `/var/mmfs/etc` on the latter. You may also choose to have different encryption configurations on different nodes; to do so, follow the steps described previously. Note the following:

- The content of `/var/mmfs/etc/RKM.conf` may be different across nodes.
- Different nodes may have different key stores.
- Installing a new encryption policy removes the previous one; as a consequence, all policy statements for the file system must be collected into a single file that can then be installed in the file system.

The scenario described here implies that the ability to create, read or write files from a particular node depends on the content of `/var/mmfs/etc/RKM.conf` and on which key stores are deployed on that node. For example, within a given cluster, read/write operations may succeed on one node and fail on another due to the fact that the right key store or `/var/mmfs/etc/RKM.conf` entry, or both, are missing.

Secure deletion

Secure deletion refers to both erasing files from the file system and erasing the MEKs that wrapped the FEKs that were used to encrypt the files.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 294

Securely deleting files in a fileset

After files have been removed from a fileset using standard file system operations (such as **unlink** and **rm**), the tenant administrator might decide to securely delete them. For example, suppose that until that point, the FEKs of all files in the fileset were encrypted with the MEK with key name `KEY-old:isklmsrv`. To cause the secure deletion of all removed files, the administrator must perform the following steps:

1. Create a new MEK and note down its key name (in this example, `KEY-old:isklmsrv`).
2. Modify the appropriate encryption policy **KEYS** statement in the encryption policy to encrypt new files with the new MEK (for example, `KEY-new:isklmsrv`) instead of the old one (`KEY-old:isklmsrv`).
3. Create and apply a migration (rewrapping) policy (**CHANGE ENCRYPTION KEYS**) to scan all files, unwrap the wrapped FEK entries of files that have been wrapped with the old key (`KEY-old:isklmsrv`), and rewrap them with the new key (`KEY-new:isklmsrv`); this step ensures that the FEKs of existing files will be accessible in the future.
4. Remove the old key, `KEY-old:isklmsrv`. This step commits the secure deletion of all files that were previously unlinked (and whose FEKs had therefore not been rewrapped with the new MEK, `KEY-new:isklmsrv`).
5. On each node that has ever done I/O to a file encrypted with the old key (`KEY-old:isklmsrv`), run the following command:

   ```
   /usr/lpp/mmfs/bin/tsctl encKeyCachePurge 'KEY-old:isklmsrv'
   ```

From this point on, the new key will be used for encryption, which will be performed transparently to the application.

Note: The **mmdelfs** command will *not* perform any secure deletion of the files in the file system to be deleted. **mmdelfs** only removes all the structures for the specified file system. To securely delete files, you need to perform the following steps:

1. Identify all MEKs currently used to wrap the FEKs of files in the file system to be deleted. If this information is not available through other means, obtain it by doing the following:
   a. Invoke **mmlsattr -n gpfs.Encryption** on all files of the file system.
   b. Parse the resulting output to extract all the distinct key names of the MEKs that are used.

   Note: These are the possible ways that an MEK might be in use in a file system:
   a. The MEK is, or was at some point, specified in an encryption rule in the policy set on the file system.
   b. An FEK rewrap has been run, rewrapping an FEK with another MEK.
2. Determine whether the identified MEKs were used to wrap FEKs in other file systems.

   WARNING: If the same MEKs were used to wrap FEKs in other file systems, deleting those MEKs will result in irreparable data loss in the other file systems where those MEKs are used. Before deleting such MEKs from the key servers, you must create one or more new MEKs and rewrap the files in the other file systems.
3. After appropriately handling any MEKs that were used to wrap FEKs in other file systems (as explained in the warning), delete the identified MEKs from their RKMs.

Secure deletion and encryption key cache purging

The key servers that store the MEKs know how to manage and securely delete keys. After an MEK is gone, all files whose FEKs were encrypted with that MEK are no longer accessible. Even if the data blocks corresponding to the deleted files are retrieved, the contents of the file can no longer be reconstructed, since the data cannot be decrypted.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 295

However, if the MEKs have been cached for performance reasons (so that they do not have to be fetched from the server each time a file is created or accessed), the MEKs must also be purged from the cache to complete the secure deletion.

You can use the following command to purge a given key from the key cache, or to clean the entire cache, of an individual node:

```
/usr/lpp/mmfs/bin/tsctl encKeyCachePurge {Key | all}
```

where:

Key
: is the key ID, specified with the *KeyId*:*RkmId* syntax.

all
: specifies that the entire key cache is to be cleaned.

The scope of this command is limited to the local node and must be run on all nodes that have accessed the MEKs you are purging in order to ensure secure deletion.

Encryption and FIPS compliance

The **FIPS1402mode** configuration variable controls whether the use of crypto-based security mechanisms (if they are to be used at all, per the GPFS administrator) is to be provided by software modules that are certified according to the requirements and standards described by the Federal Information Processing Standards (FIPS) 140 Publication Series. When in FIPS 140-2 mode, GPFS uses the FIPS 140-2 approved cryptographic provider(s); IBMJCEFIPS (certificate 376) and/or IBMJSSEFIPS (certificate 409) and/or IBM Crypto for C (ICC) (certificate 384) for cryptography. The certificates are listed on the NIST web site at http://csrc.nist.gov/cryptval/140-1/1401val2004.html.

The value of **FIPS1402mode** can be changed with the **mmchconfig** command. The default value for this variable is **no**. With **FIPS1402mode=no**, Linux nodes will use kernel encryption modules for direct I/O. If a cluster is configured with **FIPS1402mode=yes**, Linux nodes whose kernels are not running in FIPS mode will see a performance degradation when using direct I/O. The GPFS daemon on the node must be restarted in order for the new setting to take place.

Encryption and NIST compliance

Encryption always uses NIST-compliant mechanisms.

Encryption and backup/restore

GPFS will deliver all data to **mmbackup** and other external backup solutions in **cleartext** whether or not the data is encrypted in GPFS. Any backups that are taken **will not** preserve the encryption status or the encrypted content of the data. Files that are recreated upon restore will be considered for encryption status based on the policy in place on the file system at the time of the restore operation.

Encryption and snapshots

GPFS preserves the encryption status of files when they are copied into global or fileset snapshots. Global snapshot restore will restore files precisely as they are in the snapshot, including FEKs and MEKs. For details about how fileset snapshot restore functions, see the description of the **mmrestorefs** command in *GPFS: Administration and Programming Reference*.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 296

Chapter 16. Miscellaneous advanced administration topics

The following topics provide information about miscellaneous advanced administration tasks:

- "Changing IP addresses and host names"
- "Enabling a cluster for IPv6" on page 276
- "Using multiple token servers" on page 276
- "Exporting file system definitions between clusters" on page 277
- "GPFS port usage" on page 277

Changing IP addresses and host names

GPFS assumes that IP addresses and host names remain constant. In the rare event that such a change becomes necessary or is inadvertently introduced by reinstalling a node with a disk image from a different node for example, follow the steps in this topic.

If all of the nodes in the cluster are affected and all the conditions in step 2 below are met:

1. Use the **mmshutdown -a** command to stop GPFS on all nodes.
2. Using the documented procedures for the operating system, add the new host names or IP addressees, but do not remove the old ones yet. This can be achieved, for example, by creating temporary alias entries in **/etc/hosts**. Avoid rebooting the nodes until the **mmchnode** command in step 3 is executed successfully. If any of these conditions cannot be met, utilize the alternate procedure described below.
3. Use **mmchnode --daemon-interface** and **--admin-interface** to update the GPFS**mmchnode --daemon-interface**

   | **Note:** If CCR is enabled, **mmchnode --daemon-interface --admin-interface** is not allowed.
4. If the IP addresses over which the subnet attribute is defined are changed, you need to update your configuration by using the **mmchconfig** command with the **subnets** attribute.
5. Start GPFS on all nodes with **mmstartup -a**.
6. Remove the unneeded old host names and IP addresses.

If only a subset of the nodes are affected, it may be easier to make the changes using these steps:

1. Before any of the host names or IP addresses are changed:
   - Use the **mmshutdown** command to stop GPFS on all affected nodes.
   - If the host names or IP addresses of the primary or secondary GPFS cluster configuration server nodes must change, use the **mmchcluster** command to specify another node to serve as the primary or secondary GPFS cluster configuration server.
   - If the host names or IP addresses of an NSD server node must change, temporarily remove the node from being a server with the **mmchnsd** command. Then, after the node has been added back to the cluster, use the **mmchnsd** command to change the NSDs to their original configuration. Use the **mmlsnsd** command to obtain the NSD server node names.
   - Use the **mmdelnode** command to delete all affected nodes from the GPFS cluster.
2. Change the node names and IP addresses using the documented procedures for the operating system.
3. If the IP addresses over which the subnet attribute is defined are changed, you need to update your configuration by using the **mmchconfig** command with the **subnets** attribute.
4. Issue the **mmaddnode** command to restore the nodes to the GPFS cluster.
5. If necessary, use the **mmchcluster** and **mmchnsd** commands to restore the original configuration and the NSD servers.

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 297

Enabling a cluster for IPv6

For newly created clusters, if any of the specified node interfaces on the **mmcrcluster** command resolves to an IPv6 address, the cluster is automatically enabled for IPv6. For existing IPv4-based clusters, follow one of the procedures that described in this section.

If you are performing the procedure during a scheduled maintenance window and GPFS can be shut down on all of the nodes in the cluster, run the command:

```
mmchconfig enableIPv6=yes
```

After the command finishes successfully, you can start adding new nodes with IPv6 addresses.

If it is not possible to shut down GPFS on all of the nodes at the same time, run the command:

```
mmchconfig enableIPv6=prepare
```

The next step is to restart GPFS on each of the nodes so that they can pick up the new configuration setting. This can be done one node at a time when it is convenient. To verify that a particular node has been refreshed, run:

```
mmdiag --config | grep enableIPv6
```

The reported value should be 1.

Once all of the nodes have been recycled in this manner, run the command:

```
mmchconfig enableIPv6=commit
```

This command will only succeed when all GPFS daemons have been refreshed. Once this operation succeeds, you can start adding new nodes with IPv6 addresses.

To convert an existing node from an IPv4 to an IPv6 interface, use one of the procedures described in "Changing IP addresses and host names" on page 275.

Using multiple token servers

Distributed locking, allowing GPFS to maintain a consistent view of the file system, is implemented using token-based lock management. Associated with every lockable object is a token.

Before a lock on an object can be granted to a thread on a particular node, the lock manager on that node must obtain a token from the token server. The total number of token manager nodes depends on the number of manager nodes defined in the cluster.

When a file system is first mounted, the file system manager is the only token server for the file system. Once the number of external mounts exceeds one, the file system manager appoints all the other manager nodes defined in the cluster to share the token server load. Once the token state has been distributed, it remains distributed until all external mounts have gone away. The only nodes that are eligible to become token manager nodes are those designated as manager nodes.

The number of files for which tokens can be retained on a manager node is restricted by the values of the **maxFilesToCache** and **maxStatCache** configuration parameters. Distributing the tokens across multiple token manager nodes allows more tokens to be managed or retained concurrently, improving performance in situations where many lockable objects are accessed concurrently.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 298

Exporting file system definitions between clusters

You can export a GPFS file system definition from one GPFS cluster to another.

To export file system definitions between clusters, follow these steps:

1. Ensure that all disks in all GPFS file systems to be migrated are in working order by issuing the **mmlsdisk** command. Verify that the disk status is ready and availability is up. If not, correct any problems and reissue the **mmlsdisk** command before continuing.
2. Stop all user activity in the file systems.
3. Follow any local administrative backup procedures to provide for protection of your file system data in the event of a failure.
4. Cleanly unmount all affected GPFS file systems. Do not use force unmount.
5. Export the GPFS file system definitions by issuing the **mmexportfs** command. This command creates the configuration output file *ExportDataFile* with all relevant file system and disk information. Retain this file as it is required when issuing the **mmimportfs** command to import your file systems into the new cluster. Depending on whether you are exporting a single file system or all of the file systems in the cluster, issue:

   ```
   mmexportfs fileSystemName -o ExportDataFile
   ```

   or

   ```
   mmexportfs all -o ExportDataFile
   ```
6. Ensure that the file system disks from the old GPFS cluster are properly connected, and are online and available to be accessed from appropriate nodes of the new GPFS cluster.
7. To complete the movement of your file systems to the new cluster using the configuration file created in Step 5, issue one of these commands, depending on whether you are importing a single file system or all of the file systems in the cluster:

   ```
   mmimportfs fileSystemName -i ExportDataFile
   ```

   or

   ```
   mmimportfs all -i ExportDataFile
   ```

GPFS port usage

The nodes in a GPFS cluster communicate with each other using the TCP/IP protocol. The port number used by the main GPFS daemon (**mmfsd**) is controlled with the **tscTcpPort** configuration parameter. The default port number is 1191.

You can specify a different port number using the **mmchconfig** command:

```
mmchconfig tscTcpPort=PortNumber
```

When the main GPFS daemon (**mmfsd**) is not running on the primary and backup configuration server nodes, a separate service (**mmsdrserv**) is used to provide access to the configuration data to the rest of the nodes in the cluster. The port number used for this purpose is controlled with the **mmsdrservPort** parameter. By default, **mmsdrserv** uses the same port number as the one assigned to the main GPFS daemon. If you change the daemon port number, you must specify the same port number for **mmsdrserv** using the following command:

```
mmchconfig mmsdrservPort=PortNumber
```

Do not change the **mmsdrserv** port number to a number different from that of the daemon port number.

Certain commands (**mmadddisk**, **mmchmgr**, and so on) require an additional socket to be created for the duration of the command. The port numbers assigned to these temporary sockets are controlled with the **tscCmdPortRange** configuration parameter. If an explicit range is not specified, the port number is

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 299

dynamically assigned by the operating system from the range of ephemeral port numbers. If you want to restrict the range of ports used by GPFS commands, use the **mmchconfig** command:

```
mmchconfig tscCmdPortRange=LowNumber-HighNumber
```

In a remote cluster setup, if GPFS on the remote cluster is configured to use a port number other than the default, you have to specify the port number to be used with the **mmremotecluster** command:

```
mmremotecluster update ClusterName -n tcpPort=PortNumber,Node,Node...
```

Table 44 provides GPFS port usage information:

Table 44. GPFS port usage

Descriptor	Explanation
Service provider	GPFS
Service name	**mmfsd** **mmsdrserv**
Port number	1191 While executing certain commands, GPFS may need to create an additional socket, which has a dynamic port number that is assigned by the operating system.
Protocols	TCP/IP
Source port range	The source port range is chosen by the operating system on the client side.
Is the service name/number pair in the default /etc/services file shipped with AIX and Linux distributions?	See the GPFS FAQ in the IBM Cluster information center (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) or GPFS FAQ in the IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter/SSFKCN/gpfsclustersfaq.html).
Is the service name/number pair added to /etc/services by a product?	No
Binaries that listen on the ports	/usr/lpp/mmfs/bin/mmfsd /usr/lpp/mmfs/bin/mmsdrserv

Can the service be configured to use a different port?	Yes. To change the main port used by GPFS, use:
	`mmchconfig tscTcpPort=PortNumber`
	
	Note: If you change the main port (daemon port) number, you must change the **mmsdrserv** port to the same number.
	
	To change the **mmsdrserv** port number to match the daemon port number, use:
	`mmchconfig mmsdrservPort=PortNumber`
	
	To change the range of port numbers used for command execution, use:
	`mmchconfig tscCmdPortRange=LowNumber-HighNumber`
	
	To specify a port number when connecting to remote clusters, use the **mmremotecluster** command.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 300

Table 44. GPFS port usage (continued)

Descriptor	Explanation
When is the service required? What depends on the service?	On the GPFS primary and secondary cluster configuration servers, either **mmsdrserv** or **mmfsd** needs to be running at all times to provide access to GPFS configuration data to the rest of the cluster. On other nodes, **mmfsd** must be running in order to mount a GPFS file system. Depending on the GPFS configuration, a node either has to be a member of the GPFS cluster or possess an authorized SSL key in order to establish a connection.
When the daemon starts and its port is already in use (for example, another resource has bound to it already), how does the daemon behave?	The daemon shuts down and tries to start over again. Most GPFS daemon down error messages are in the **mmfs.log.previous** log for the instance that failed. If the daemon restarted, it generates a new **mmfs.log.latest** log. Begin problem determination for these errors by examining the operating system error log. GPFS records file system or disk failures using the error logging facility provided by the operating system: **syslog** facility on Linux and **errpt** facility on AIX. See the *GPFS: Problem Determination Guide* for further information.
Is there an administrator interface to query the daemon and have it report its port number?	Yes; run this command: **mmlsconfig tscTcpPort**

Is the service/port registered with the Internet Assigned Numbers Authority (IANA)?	Yes
	`gpfs 1191/tcp General Parallel File System`
	`gpfs 1191/udp General Parallel File System`
	`# Dave Craft <gpfs@ibm.com>`
	`November 2004`

Note: Ports configured for **gpfsClusterRemoteShellCommand** (for example, **ssh**) and ICMP (ping) also must be unblocked in the firewall for GPFS to function properly

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 301

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 302

Accessibility features for GPFS

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 GPFS:

- 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

The **IBM Cluster Information Center**, and its related publications, are accessibility-enabled. The accessibility features of the information center are described in the Accessibility topic at the following URL: http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.addinfo.doc/access.html.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility:

`http://www.ibm.com/able/`

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 303

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 304

Notices

This information was developed for products and services offered in the U.S.A.

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
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (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 Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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 Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 305

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. 30ZA/Building 707
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment or 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.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 306

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, Windows, and Windows NT 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.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 307

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 308

Glossary

This glossary provides terms and definitions for the GPFS product.

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 (http://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 is the node with the lowest node number among the quorum nodes that are operating at a particular time.

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*.

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.

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.

domain
A logical grouping of resources in a network for the purpose of common management and administration.

© Copyright IBM Corporation © IBM 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 309

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 File encryption key. An FEK is used to encrypt sectors of an individual file.

fileset A hierarchical grouping of files managed as a unit for balancing workload across a cluster. See also *dependent fileset* and *independent fileset*.

fileset snapshot
A snapshot of an independent fileset plus all dependent filesets.

file clone
A writable snapshot of an individual file.

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.

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 310

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

MEK Master encryption key. An MEK is used to encrypt other keys.

metadata
A data structures that contain access information about file data. These include: inodes, indirect blocks, and directories. These data structures are 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.

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 311

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.

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
Remote key management server. An RKM server is used to store MEKs.

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

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 312

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* and *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

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 an virtual file system (VFS).

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 313

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 314

Index

Special characters

/var/mmfs/etc/RKM.conf files 267

A

accessibility features for the GPFS product 281
active file management, restrictions in 245
adding
  components 173
adminDrain, pdisk state 164
administration, clustered NFS subsystem 97
administration, collector node 139
AFM and mmbackup 242
AFM, restrictions in 245
appendOnly 58
array, declustered 158, 165
  background tasks 170
  large 166
  managing 161
  parameters 166
  size 166
  small 166
  spare space 166
attributes
  component, updating 176
attributes, using file 26
auto-generated ID mappings
  Windows 149

B

backend, RKM 267
backends, RKM 268
background tasks 170
backup
  storage pools 50
backup/restore and encryption 274
block size 167
built-in functions 29

C

cabling (HVA), disk enclosure 212
cache purging, encryption key 273
callbacks 181
  daRebuildFailed 181
  nsdCksumMismatch 181
  pdFailed 181
  pdPathDown 181
  pdReplacePdisk 181
  postRGRelinquish 181
  postRGTakeover 181
  preRGRelinquish 181
  preRGTakeover 181
  rgOpenFailed 181
  rgPanic 181
changing
  hostnames 275
  IP addresses 275
changing *(continued)*
  node names 275
  node numbers 275
CHAR 33
checksum
  data 171
  end-to-end 155
child fileset 52
cipherlist 8
clause
  DIRECTORIES_PLUS 22
  EXCLUDE 22
  FOR FILESET 23
  FROM POOL 23
  GROUP POOL 23
  LIMIT 23
  REPLICATE 24
  SET POOL 24
  SHOW 24
  THRESHOLD 24
  TO POOL 25
  WEIGHT 25
  WHEN 26
  WHERE 26
clauses
  syntax definitions 22
clauses for policy rules 20
clones, file
  creating 67
  deleting 69
  disk space management 69
  listing 68
  management 67
  policy files 70
  snapshots 69
  splitting from clone parents 69
cluster configuration 173, 174
cluster configuration information, SNMP 142
cluster status information, SNMP 141
clustered NFS set up 96
clustered NFS subsystem
  using 95
clustered NFS subsystem, administration 97
clusters
  accessing file systems 1
collector node administration 139
collector node, installing MIB files 139
commands
  mmaddcallback 181
  mmadddisk 16
  mmaddnode 275
  mmapplypolicy 16, 19, 36, 37, 39, 44, 64
  mmauth 1, 3, 4, 6, 8, 9, 10
  mmbackup 54
  mmchattr 16
  mmchcluster 275
  mmchconfig 1, 4, 6, 11, 95, 275, 276
  mmchdisk 16
  mmchfileset 57
  mmchnode 275
  mmchnsd 275

© Copyright IBM Corp. 2014

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 315

commands *(continued)*
mmchpolicy 44
mmcrcluster 1
mmcrfileset 55
mmcrfs 16
mmcrsnapshot 61, 63
mmdeldisk 16, 17
mmdelfileset 56
mmdelnode 275
mmdelsnapshot 65
mmdf 10, 14, 15, 17
mmexportfs 277
mmfsck 10
mmimportfs 277
mmlinkfileset 52, 55, 56, 57
mmlsattr 17, 19, 52, 57
mmlscluster 5
mmlsconfig 11
mmlsdisk 10, 277
mmlsfileset 19, 53, 54, 56, 57
mmlsfs 10, 17
mmlsmount 10
mmlsnsd 275
mmlspolicy 45
mmlssnapshot 63
mmmount 5
mmpmon 103
mmremotecluster 1, 5, 9, 10
mmremotefs 1, 5, 10
mmrestorefs 54, 63
mmrestripefile 16, 18
mmrestripefs 16, 18, 39
mmshutdown 275
mmsnapdir 54, 64
mmstartup 275
mmunlinkfileset 52, 56, 57
component
adding 173
configuration 172
defining locations 174
component attributes
updating 176
components of storage enclosures
replacing failed 224
CONCAT 33
configuration
of components 172
configuration example
GPFS Native RAID on IBM System x GPFS Storage Server (GSS) 201
configuring
GPFS nodes 205
configuring ID mappings in IMU
Windows 150
configuring Net-SNMP 138
configuring SNMP-based management applications 139
creating
recovery groups 187
recovery groups on IBM System x GPFS Storage Server (GSS) 205
creating and defining vdisks 215
CURRENT_DATE 34
CURRENT_TIMESTAMP 26, 34

D

daRebuildFailed callback 181
data checksum 171
data deletion, secure 261
data management
policy-based for GPFS 13
data protection 261
data redundancy 154
date and time functions 34
DAY 35
DAYOFWEEK 34
DAYOFYEAR 34
DAYS 35
DAYSINMONTH 35
DAYSINYEAR 35
dead, pdisk state 164
declustered array 158, 165
background tasks 170
large 166
managing 161
parameters 166
size 166
small 166
spare space 166
declustered array free space 166
declustered RAID 155
default encryption value 264
defining
component locations 174
external pools 45
defining and creating vdisks 215
defining the recovery group layout 213
DELETE rule 19, 25, 36
deletion of data, secure 261
diagnosing, pdisk state 164
diagnosis, disk 169
DIRECTORIES_PLUS 22
disaster recovery
active/active PPRC-based mirrored GPFS cluster
data integrity 89
description 82
PPRC consistency groups 89
setting up a configuration 83
steps after a disaster 84
active/passive PPRC-based mirrored GPFS cluster
data integrity 89
description 86
PPRC consistency groups 89
setting up a configuration 86
steps after a disaster 88
asynchronous mirroring utilizing ESS FlashCopy
description 90
description 75
failback 80
failover 79
synchronous mirroring utilizing GPFS replication
description 76
setting up 77
steps after a disaster 78
synchronous mirroring utilizing IBM TotalStorage Enterprise Storage Server (ESS) Peer-to-Peer Remote Copy (PPRC)
description 82
disk configuration information, SNMP 146
disk descriptor 16
disk enclosure and HBA cabling 212
disk performance information, SNMP 146

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 316

disk status information, SNMP 145
disks
  configuration 157, 165
    declustered array 158
    recovery group 157
  declustered array 165
  diagnosis 169
  hardware service 172
  hospital 160, 169
  maintenance 169
  pdisk 159, 162
  pdisk free space 167
  pdisk paths 163
  pdisk states 164
  physical 159, 162
  replacement 160, 171
  replacing failed 194, 218
  setup example 182
  solid-state 160
  spare space 166
  SSD 160
  storage pool assignment 16
  vdisk 159, 167
  vdisk size 167
  virtual 159, 167
display IDs
  synchronizing 175
displaying
  vdisk I/O statistics 180
distributedTokenServer 276
drives and enclosures, updating firmware on 200

E

EINVAL 20, 44
enabling cluster for IPv6
  IPv6, enabling a cluster for 276
enclosure (disk) and HBA cabling 212
enclosure components
  replacing failed 224
enclosures and drives, updating firmware on 200
encryption 261
encryption and backup/restore 274
encryption and FIPS compliance 274
encryption and NIST compliance 274
encryption and snapshots 274
ENCRYPTION IS policy rule 262
encryption key cache purging 273
encryption keys 261
encryption policies 262
encryption policies, rewrapping 266
encryption policy example 265
encryption policy rules 262
encryption setup requirements 267
encryption use-case scenarios 269
encryption value, default 264
encryption-enabled environment 269
ENCRYPTION, SET (policy rule) 262
end-to-end checksum 155
environment, encryption-enabled 269
events
  LOW_SPACE and NO_SPACE 50
example of encryption policy 265
examples
  configuring GPFS Native RAID 201
  creating recovery groups 187
examples *(continued)*
  creating recovery groups on IBM System x GPFS Storage Server (GSS) 205
  disk setup 182
  policies and rules 39
  preparing recovery group servers 182, 201
EXCLUDE 22
EXCLUDE rule 19
expressions, SQL 26
extended attribute functions
  XATTR_FLOAT() 30
  XATTR_INTEGER() 30
  XATTR() 30
external lists 51
external pools
  defining 45
  managing 46
  migrate 48
  pre-migrating files 49
  purging files 49
  recall 48
  requirements 19
  using 45, 48, 49
  using thresholds 50
external storage pools 18
  working with 45

F

failed disks, replacing 194, 218
failed enclosure components, replacing 224
failing, pdisk state 164
failover, NFS 95
failover, server 171
features, GPFS Native RAID 153, 154
FEKs 261
file attributes, using 26
file clones
  creating 67
  deleting 69
  disk space management 69
  listing 68
  management 67
  policy files 70
  snapshots 69
  splitting from clone parents 69
file encryption keys 261
file list format 47
file management policies 19
File Placement Optimizer 255
file placement policies 19
file system performance information, SNMP 144
file system status information, SNMP 143
file systems 4
  access from other cluster 1
  additional information 10
  administration 10
  coordinating changes 11
  displaying information 11
  displaying statistics 104, 106
  exporting 277
  granting access 6
  managing remote access 6
  multiple security levels 8
  physical connection 1
  record format 47
  remote access 6

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 317

file systems *(continued)*
remote mount 4
remote mount concepts 1
reset statistics 107
restoring from a snapshot 63
revoking access 6
security keys 9, 10
user access 3
virtual connection 1
file, list format 47
files
/etc/group 3
/etc/passwd 3
/var/mmfs/ssl/id_rsa.pub 4, 9
storage pool assignment 16
files, purging
external pools 49
filesets
attributes 57
changing 57
creating 55
deleting 56
in global snapshots 53
linking 56
managing 55
namespace attachment 52, 53
overview 52
quotas 53
renaming 57
root 56, 57
setting and changing the immutability of files 58
snapshots 54
status 57
storage pool usage 53
unlinking 57
with mmbackup 54
FIPS compliance and encryption 274
FIPS1402mode 274
firmware, updating on enclosures and drives 200
FOR FILESET 23
format
file list 47
formatting, pdisk state 164
FPO 255
free space, declustered array 166
free space, pdisk 167
FROM POOL 23
functions
CHAR 33
CONCAT 33
CURRENT_DATE 34
CURRENT_TIMESTAMP 34
DAY 35
DAYOFWEEK 34
DAYOFYEAR 34
DAYS 35
DAYSINMONTH 35
DAYSINYEAR 35
HEX 33
HOUR 35
INT 34
INTEGER 34
LENGTH 33
LOWER 33
MINUTE 35
MOD 34
MONTH 35
functions *(continued)*
QUARTER 35
REGEX 33
SECOND 35
SUBSTR 34
SUBSTRING 34
TIMESTAMP 35
UPPER 34
VARCHAR 34
WEEK 35
YEAR 35

G

global namespace 14, 15
global snapshots
creating 61
deleting 65
file clones 69
linking 64
listing 63
overview 61
restoring file systems 63
using policy 64
with filesets 53
GNR 153
GPFS Native RAID
callbacks 181
configuring on IBM System x GPFS Storage Server (GSS) 201
data redundancy 154
declustered array 158
declustered RAID 155
disk configuration 157
disk hospital 160
disk replacement 160
end-to-end checksum 155
features 153, 154
health metrics 160
introduction 153
managing 161
monitoring 179
overview 153
pdisk 159
pdisk discovery 160
physical disk 159
planning considerations 178
RAID code 154, 167
recovery group 157
recovery group server parameters 161
solid-state disk 160
SSD 160
system management 178
vdisk 159
virtual disk 159
GPFS nodes
configuring 205
GPFS port usage 277
GPFS recovery, NFS failover 95
GPFS Storage Server 172, 173, 174, 175, 176
GPFS support for SNMP 137
group ID 3
GROUP POOL 23
group, recovery 157
creating 162, 187
creating on IBM System x GPFS Storage Server (GSS) 205
log vdisks 167

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 318

group, recovery *(continued)*
managing 161
overview 161
server failover 162
server parameters 161
verifying 190, 208
GSS 212
configuring GPFS Native RAID on 201

H

hardware service 172
HBA cabling, disk enclosure 212
health metrics 160
HEX 33
Hierarchical Storage Management 18, 19
high threshold 49
high-availability configuration 268
HighPercentage 25
hospital, disk 160, 169
hostnames, changing 275
HOUR 35
HSM 18, 19

I

IBM System x GPFS Storage Server (GSS) 212
configuring GPFS Native RAID on 201
creating recovery groups on 205
identity management on Windows 149
IDs
display, synchronizing 175
ILM 13, 18, 48, 49, 50
image backup 71
image restore 71
implementing
external pool management 46
Information Lifecycle Management 13, 18, 48, 49, 50
init, pdisk state 164
installing GPFS, using mksysb 275
installing MIB files on the collector and management node, SNMP 139
installing Net-SNMP 137
installing Windows IMU 149
INT 34
INTEGER 34
internal storage pools 14
introduction, GPFS Native RAID 153
IP addresses, changing 275

J

job 36
junction 52
junction path 56

K

key cache purging, encryption 273
keys, encryption 261

L

large declustered array 166
LENGTH 33
license inquiries 283
LIMIT 23
listing information
vdisk I/O statistics 180
lists, external 51
locations of components
defining 174
locking and load balancing, NFS 96
log vdisks 167
LogTip NVRAM partitions 212
low threshold 49
LOW_SPACE even 50
low-occupancy-percentage 25
LOWER 33

M

m4 43
maintenance, disk 169
management
external pools 46
management and monitoring, SNMP subagent 140
management node, installing MIB files 139
management, system 178
managing
filesets 55
managing GPFS Native RAID 161
master encryption keys 261
maxblocksize 11
maxFilesToCache 276
maxStatCache 276
MDS 229
MEKs 261
metadata server 229
MIB files, installing on the collector and management node 139
MIB objects, SNMP 141
migrate
external pools 48
MIGRATE rule 19, 25, 36
MINUTE 35
miscellaneous SQL functions 36
missing, pdisk state 164
mmaddcallback 181
mmapplypolicy 49
mmbackup and AFM 242
mmexportfs 277
mmimportfs 277
mmpmon
adding nodes to a node list 108
concurrent processing 104
counters 135
delete nodes from node list 111
deleting a node list 109
disable 118
display node list 110
display statistics 123
enable 119
examples 126
fs_io_s 104
histogram reset 122
input 103
interpreting results 126
interpreting rhist results 128
io_s 106
latency ranges 116
miscellaneous information 134

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 319

mmpmon *(continued)*
multiple node processing 104
new node list 110
nlist 107
nlist add 108
nlist del 109
nlist failures 114
nlist new 110
nlist show 110
nlist sub 111
node list facility 107, 112
node list failure values 114
once 129
output considerations 134
overview 103
pattern 120
request histogram 115
reset 107
return codes 135
rhist nr 117
rhist off 118
rhist on 119
rhist p 120
rhist reset 122
rhist s 123
size ranges 115
source 129
specifying new ranges 117
version 125
mmpmon command input 180
MOD 34
monitoring
health 99
network performance 99
performance 99
system 179
monitoring, NFS 95
MONTH 35
multicluster
file system access 1

N

namespace
global 14, 15
Net-SNMP
configuring 138
installing 137
running under SNMP master agent 140
traps 147
network performance 99
NFS failover 95
NFS locking and load balancing 96
NFS monitoring utility 95
NIST compliance 10
NIST compliance and encryption 274
NO_SPACE events 50
noData, pdisk state 164
node configuration information, SNMP 142
node numbers, changing 275
node status information, SNMP 142
nodes
configuring 205
renaming or renumbering 275
noPath, pdisk state 164
noRGD, pdisk state 164
notices 283
noVCD, pdisk state 164
NSD server 1
nsdCksumMismatch callback 181
numerical functions 34

O

object IDs
SNMP 141
ok, pdisk state 164
overview, GPFS Native RAID 153

P

partitions, LogTip NVRAM 212
patent information 283
paths, pdisk 163
pdFailed callback 181
pdisk free space 167
pdisks 159, 162
discovery 160
health metrics 160
managing 161
overview 162
paths 163
states 164
pdPathDown callback 181
pdReplacePdisk callback 181
performance
monitoring 99, 103
network 99
physical disk 159, 162
planning considerations 178
policies
definition 19
file placement 53
policies and rules
built-in functions 29
changing the active policy 45
commands
mmapplypolicy 64
mmchpolicy 44, 45
conditions identified 20
creating a policy 44
date and time functions 34
default 45
deleting policies 45
examples and tips 39
installing a policy 44
introduction 19
listing policies 45
macro processing 43
managing policies 44
mmchpolicy 45
numerical functions 34
string functions 33
syntax definitions 22
validating policies 45
policies, encryption 262
policies, rewrapping 266
policy example, encryption 265
policy rule, ENCRYPTION IS 262
policy rule, SET ENCRYPTION 262
policy rules
clauses 20
SQL expressions 26

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 320

policy rules *(continued)*
types 20
policy rules, encryption 262
pools
using external 18
using internal 14
working with external storage 45
pools, external
defining 45
HSM 49
managing 46
purging files 49
requirements 19
using 48, 49
pools, storage
backup and restore 50
port usage, GPFS 277
postRGRelinquish callback 181
postRGTakeover callback 181
pre-migrate threshold 49
pre-migrating files
external pools 49
preparing recovery group servers 182
configuring GPFS Native RAID on GSS 201
preRGRelinquish callback 181
preRGTakeover callback 181
private IP address 6
protection of data 261
PTOW, pdisk state 164
public IP address 6
purging files
external pools 49
purging, encryption key cache 273

Q

QUARTER 35

R

RAID code
comparison 154
planning considerations 178
Reed-Solomon 154
replication 154
vdisk configuration 167
RAID layouts
conventional 155
declustered 155
RAID, declustered 155
readonly, pdisk state 164
rebalance, background task 170
rebuild-1r, background task 170
rebuild-2r, background task 170
rebuild-critical, background task 170
rebuild-offline, background task 170
recall
external pools 48
record format 47
recovery group creation, verifying 214
recovery group layout, defining 213
recovery group servers, preparing 182, 201
recovery groups 157
configuring 187, 205
creating 162, 187
creating on IBM System x GPFS Storage Server (GSS) 205
recovery groups *(continued)*
layout 188, 206
log vdisks 167
managing 161
overview 161
preparing servers 182
server failover 162, 171
server parameters 161
stanza files 188, 206
verifying 190, 208
recovery groups GSS
configuring GPFS Native RAID on 201
preparing servers 201
redundancy codes
comparison 154
Reed-Solomon 154
replication 154
REGEX 33
remote key management server setup 267
repair-RGD/VCD, background task 170
replace, pdisk state 164
replacement, disk 160, 171
replacing failed disks 194, 218
replacing failed storage enclosure components 224
REPLICATE 24
REPLICATE clause 24
requirements
external pools 19
requirements (setup), encryption 267
restore
storage pools 50
restore/backup and encryption 274
restrictions in active file management 245
rewrapping policies 266
rgOpenFailed callback 181
rgPanic callback 181
RKM backend 267
RKM backends 268
RKM server setup 267
RKM.conf files 267
root fileset 56, 57
root squash 3
rule (policy), ENCRYPTION IS 262
rule (policy), SET ENCRYPTION 262
RULE clause
DIRECTORIES_PLUS 22
EXCLUDE 22
FOR FILESET 23
FROM POOL 23
GROUP POOL 23
LIMIT 23
REPLICATE 24
SET POOL 24
SHOW 24
THRESHOLD 24
TO POOL 25
WEIGHT 25
WHEN 26
WHERE 26
rules
policy 20
rules, encryption policy 262
rules, policy
SQL expressions 26

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 321

S

Scale Out Backup and Restore 71
  backup procedure 71
  restore procedure 73
scenarios (use-case), encryption 269
scrub, background task 170
SECOND 35
secure deletion of data 261
security key
  changing 9, 10
server failover 171
server setup, RKM 267
server, metadata 229
service, hardware 172
SET ENCRYPTION policy rule 262
SET POOL 24
setting and changing the immutability of files 58
  effects of file operations on immutable files 59
  immutability restrictions 58
setup example, disk 182
setup requirements, encryption 267
setup, RKM server 267
SHOW 24
size, vdisk 167
small declustered array 166
SNAP_ID 36
snapshots
  viewing 240
snapshots and encryption 274
snapshots, fileset 54
snapshots, global
  creating 61
  deleting 65
  file clones 69
  linking 64
  listing 63
  overview 61
  restoring file systems 63
  using policy 64
  with filesets 53
SNMP
  cluster configuration information 142
  cluster status information 141
  collector node administration 139
  configuring Net-SNMP 138
  configuring SNMP-based management applications 139
  disk configuration information 146
  disk performance information 146
  disk status information 145
  file system performance information 144
  file system status information 143
  GPFS support 137
  installing MIB files on the collector and management node 139
  installing Net-SNMP 137
  management and monitoring subagent 140
  MIB objects 141
  Net-SNMP traps 147
  node configuration information 142
  node status information 142
  object IDs 141
  starting and stopping the SNMP subagent 140
  storage pool information 145
SOBAR 71
  backup procedure 71
  restore procedure 73
solid-state disk 160
spare space 166
SQL expressions 26
SQL functions
  SNAP_ID 36
SQL functions, miscellaneous
  functions, miscellaneous SQL 36
SSD 160
stanza files 16
  recovery group 188, 206
  vdisk 191, 209
states, pdisk 164
states, vdisk 168
statistics, vdisk I/O
  displaying 180
  resetting 180
storage management
  policy-based for GPFS 13
storage pool information, SNMP 145
storage pools
  backup 50
  changing a disk's assignment 16
  creating 16
  deleting 16
  file assignment 16
  introduction 13
  listing disks in 17
  listing pools for a file system 17
  listing the pool for a file 17
  managing 15
  rebalancing 18
  replication 18
  restore 50
  system storage pool 14
  system.log storage pool 14
  user storage pools 15
  using external 18
  using internal 14
  working with external 45
storage, tiered
  external storage pools 18
string functions 33
subnet 6
subnets 275
SUBSTR 34
SUBSTRING 34
suspended, pdisk state 164
synchronizing
  display IDs 175
syntax definitions
  clauses 22
  policies and rules 22
system
  management 178
  monitoring 179
  planning 178
system storage pool 16, 19, 20
  definition 14
system.log storage pool
  definition 14
systemDrain, pdisk state 164

T

tasks, background 170
THRESHOLD 24
thresholds
  high 49

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 322

thresholds *(continued)*
low 49
pre-migrate 49
tiered storage
external storage pools 18
TIMESTAMP 35
Tivoli Storage Manager 19
TO POOL 25
trademarks 284
traps, Net-SNMP 147
TSM 19
types
clauses for policy rules 20
policy rules 20

U

UID remapping 3
updating
component attributes 176
updating firmware on enclosures and drives 200
UPPER 34
use-case scenarios, encryption 269
user account 3
user ID 3
user provided program for managing external pools 46
using
external pools 45, 48, 49
external storage pools 18
internal storage pools 14
using a clustered CNFS subsystem
network setup 96
using a clustered NFS subsystem 95
administration 97
clustered NFS set up 96
setup 96
using file attributes 26
using the policy engine to read a snapshot 64
using thresholds with external pools 50
utility, NFS monitoring 95

V

VARCHAR 34
vdisks 159, 167
block size 167
creating 191, 209
creating NSDs 193, 211
data checksum 171
defining 191, 209
I/O statistics, displaying 180
log vdisks 167
managing 161
RAID code 167
relationship with NSDs 168
size 167
stanza files 191, 209
states 168
vdisks, defining and creating 215
verifying recovery group creation 214
viewing snapshots 240
virtual disk 159, 167

W

WEEK 35
WEIGHT 25, 49
WHEN 26
WHERE 26
Windows
auto-generated ID mappings 149
configuring ID mappings in IMU 150
identity management 149
IMU installation 149
installing IMU 149
working with external storage pools 45

Y

YEAR 35

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 323

Meta Platforms, Inc. and Samsung Electronics Co. Ltd. - Ex. 1032
Page 324



Product Number: 5725-Q01
5641-GPF
5641-GP6
5641-GP7
5641-GP8

Printed in USA

SC23-7032-00

