AIX Clusters Concepts, Planning, and Installation Guide
AIX Clusters Concepts, Planning, and Installation Guide
# Contents

**Figures** ....................................................................................................................... vii

**About This Book** ........................................................................................................... ix
Who Should Use This Book ................................................................................................ ix
How this book is organized ................................................................................................. ix
  Typography and Terminology ........................................................................................... x

**What’s new** .................................................................................................................... xi
What’s new for GPFS 2.1 ...................................................................................................... xi
Migration .............................................................................................................................. xii

**Part 1. Understanding GPFS.** ......................................................................................... 1

**Chapter 1. Introducing General Parallel File System** ..................................................... 3
The strengths of GPFS ........................................................................................................ 3
  Improved system performance ......................................................................................... 3
  Assured file consistency ................................................................................................ 3
  High recoverability and increased data availability ...................................................... 3
  Enhanced system flexibility ........................................................................................... 4
  Simplified administration .............................................................................................. 4
The basic GPFS structure ................................................................................................... 5
  The GPFS kernel extension ........................................................................................... 5
  The GPFS daemon ......................................................................................................... 6
The AIX cluster environment .............................................................................................. 6
  The RSCT peer domain environment ........................................................................... 7
  The HACMP environment ............................................................................................ 7

**Chapter 2. Planning for GPFS.** ...................................................................................... 9
Hardware specifications ..................................................................................................... 9
Programming specifications ............................................................................................... 9
Recoverability considerations ............................................................................................ 9
  Node failure .................................................................................................................. 10
  Disk failure .................................................................................................................. 10
  Making your decision ................................................................................................... 10
Disk considerations .......................................................................................................... 11
  Disk fencing ................................................................................................................ 11
Logical volume creation considerations ............................................................................ 11
GPFS cluster creation considerations ............................................................................... 14
  Nodes in your GPFS cluster ....................................................................................... 14
  GPFS cluster data servers ......................................................................................... 14
  GPFS cluster type ........................................................................................................ 15
  Remote shell command ............................................................................................... 15
  Remote file copy command ......................................................................................... 15
Nodeset configuration considerations ............................................................................... 15
  Nodes in your GPFS nodeset ....................................................................................... 16
  Nodeset identifier ........................................................................................................ 17
  The operation of nodes ............................................................................................... 17
  Maximum file system block size allowed .................................................................... 19
  DMAP interface options ............................................................................................... 19
A sample nodeset configuration ......................................................................................... 19
File system creation considerations .................................................................................. 20
  Automatic mount ......................................................................................................... 21
  Estimated node count ................................................................................................. 21
File system sizing .................................................. 21
File system recoverability parameters .......................... 22
Automatic quota activation ........................................ 23
Disk verification .................................................... 24
Enable DMAPI ....................................................... 24
Mountpoint directory .............................................. 24
Device name of the file system .................................. 25
Disks for the file system ........................................... 25
Nodeset to which the file system belongs ......................... 25
A sample file system creation .................................... 26

Part 2. Preparing your system for GPFS .......................... 27

Chapter 3. Installing GPFS .......................................... 29
Electronic license agreement ...................................... 29
Files to ease the installation process ............................ 29
Verify there is no conflicting software installed ................. 29
Verifying the level of prerequisite software ....................... 30
Installation procedures ............................................. 31
Creating the GPFS directory ...................................... 31
Installing the GPFS man pages .................................... 31
Creating the GPFS installation images ........................... 32
Installing GPFS on your network .................................. 32
Verifying the GPFS installation .................................... 33
What’s next after completing the installation of GPFS ............ 33

Chapter 4. Tuning your system for GPFS ........................... 35
System configuration settings ...................................... 35
Security ............................................................. 35
Topology Services .................................................. 35
Communications I/O ............................................... 35
Disk I/O ............................................................ 36
nofiles ............................................................ 36
MANPATH environment variable ................................. 36

Chapter 5. Migration, coexistence, and compatibility ............ 37
Migrating to GPFS 2.1 .............................................. 37
GPFS nodesets for migration ...................................... 37
Staged migration to GPFS 2.1 .................................... 37
Full migration to GPFS 2.1 ........................................ 38
Reverting to the previous level of GPFS .......................... 39
Coexistence ......................................................... 39
Compatibility ....................................................... 40

Chapter 6. Permanently uninstalling GPFS .......................... 41

Part 3. Appendixes ................................................... 43

Appendix A. GPFS architecture ..................................... 45
Special management functions .................................... 45
The GPFS configuration manager ................................. 45
The file system manager .......................................... 45
The metanode ....................................................... 47
Use of disk storage and file structure within a GPFS file system 47
Metadata ........................................................... 47
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quota files</td>
<td>49</td>
</tr>
<tr>
<td>Log files</td>
<td>49</td>
</tr>
<tr>
<td>User data</td>
<td>49</td>
</tr>
<tr>
<td>GPFS and memory</td>
<td>49</td>
</tr>
<tr>
<td>Component interfaces</td>
<td>50</td>
</tr>
<tr>
<td>Program interfaces</td>
<td>50</td>
</tr>
<tr>
<td>Socket communications</td>
<td>51</td>
</tr>
<tr>
<td>Application and user interaction with GPFS</td>
<td>52</td>
</tr>
<tr>
<td>Operating system commands</td>
<td>52</td>
</tr>
<tr>
<td>Operating system calls</td>
<td>53</td>
</tr>
<tr>
<td>GPFS command processing</td>
<td>56</td>
</tr>
<tr>
<td>Recovery</td>
<td>57</td>
</tr>
<tr>
<td>GPFS cluster data</td>
<td>57</td>
</tr>
<tr>
<td>Appendix B. Considerations for GPFS applications</td>
<td>59</td>
</tr>
<tr>
<td>Exceptions to Open Group technical standards</td>
<td>59</td>
</tr>
<tr>
<td>Application support</td>
<td>59</td>
</tr>
<tr>
<td>Appendix C. Restrictions and conventions for GPFS</td>
<td>61</td>
</tr>
<tr>
<td>GPFS cluster configuration</td>
<td>61</td>
</tr>
<tr>
<td>GPFS nodeset configuration</td>
<td>62</td>
</tr>
<tr>
<td>Starting GPFS</td>
<td>62</td>
</tr>
<tr>
<td>GPFS file system configuration</td>
<td>63</td>
</tr>
<tr>
<td>GPFS cluster administration</td>
<td>63</td>
</tr>
<tr>
<td>GPFS nodeset administration</td>
<td>64</td>
</tr>
<tr>
<td>GPFS file system administration</td>
<td>64</td>
</tr>
<tr>
<td>Disk administration in your GPFS file system</td>
<td>66</td>
</tr>
<tr>
<td>Communicating file accessing patterns</td>
<td>67</td>
</tr>
<tr>
<td>System configuration</td>
<td>68</td>
</tr>
<tr>
<td>Notices</td>
<td>69</td>
</tr>
<tr>
<td>Trademarks</td>
<td>70</td>
</tr>
<tr>
<td>Glossary</td>
<td>73</td>
</tr>
<tr>
<td>Bibliography</td>
<td>77</td>
</tr>
<tr>
<td>GPFS publications</td>
<td>77</td>
</tr>
<tr>
<td>AIX publications</td>
<td>77</td>
</tr>
<tr>
<td>Reliable Scalable Cluster Technology publications</td>
<td>78</td>
</tr>
<tr>
<td>HACMP/ES publications</td>
<td>78</td>
</tr>
<tr>
<td>Storage related information</td>
<td>78</td>
</tr>
<tr>
<td>Redbooks</td>
<td>78</td>
</tr>
<tr>
<td>Whitepapers</td>
<td>78</td>
</tr>
<tr>
<td>Non-IBM publications</td>
<td>79</td>
</tr>
<tr>
<td>Index</td>
<td>81</td>
</tr>
</tbody>
</table>
# Figures

1. An RSCT peer domain environment ........................................... 7
2. An HACMP environment ......................................................... 8
3. RAID/ESS Controller multi-tailed to each node ........................... 10
4. GPFS files have a typical UNIX structure ................................... 48
About This Book

The General Parallel File System for AIX 5L: AIX Clusters Concepts, Planning, and Installation Guide describes:

- The IBM General Parallel File System (GPFS) licensed program.
- Planning concepts for GPFS.
- The installation and migration of GPFS.
- Tuning your system for GPFS.

Throughout this publication you will see various command and component names beginning with the prefix mmfs. This is not an error. GPFS shares many components with the related products IBM Multi-Media Server and IBM Video Charger. Consequently, the coexistence of GPFS with either the IBM Multi-Media Server product or the IBM Video Charger product is not supported. See "Verify there is no conflicting software installed" on page 29.

Who Should Use This Book

This book is intended for system administrators, analysts, installers, planners, and programmers of GPFS systems. It assumes that you are, and it is particularly important that you be, experienced with and understand the AIX 5L™ operating system and the subsystems used to manage disks. For an RSCT peer domain environment, it also assumes that you are experienced with and understand the RSCT subsystem of AIX 5L and the management of peer domains. For an HACMP environment, it also assumes that you are experienced with and understand the High Availability Cluster Multi-Processing for AIX Enhanced Scalability (HACMP/ES) program product and the subsystems used to manage disks. Use this book if you are:

- Planning for GPFS
- Installing GPFS
- Migrating to the latest level of GPFS
- Tuning your environment for GPFS

For a list of related books you should be familiar with, see the "Bibliography" on page 77.

How this book is organized

Part 1, “Understanding GPFS” includes:

- Chapter 1, “Introducing General Parallel File System”, on page 3
- Chapter 2, “Planning for GPFS”, on page 9

Part 2, “Preparing your system for GPFS” includes:

- Chapter 3, “Installing GPFS”, on page 29
- Chapter 4, “Tuning your system for GPFS”, on page 35
- Chapter 5, “Migration, coexistence, and compatibility”, on page 37

The Appendixes includes:

- Appendix A, “GPFS architecture”, on page 45
- Appendix B, “Considerations for GPFS applications”, on page 59
- Appendix C, “Restrictions and conventions for GPFS”, on page 61

"Notices" on page 69
## Typography and Terminology

This book uses the following typographical conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold</strong></td>
<td><strong>Bold</strong> words or characters represent system elements that you must use literally, such as commands, subcommands, flags, path names, directories, file names, values, and selected menu options.</td>
</tr>
<tr>
<td><strong>Bold Underlined</strong></td>
<td><strong>Bold Underlined</strong> keywords are defaults. These take effect if you fail to specify a different keyword.</td>
</tr>
</tbody>
</table>
| *Italic*   | *Italic* words or characters represent variable values that you must supply  
|            | *Italics are used for book titles  
|            | *Italics are used for general emphasis |
| **Monospace** | All of the following are displayed in monospace type:  
|            | *Displayed information  
|            | *Message text  
|            | *Example text  
|            | *Specified text typed by the user  
|            | *Field names as displayed on the screen  
|            | *Prompts from the system  
|            | *References to example text |

**[]** Brackets enclose optional items in format and syntax descriptions.

**{}** Braces enclose a list from which you must choose an item in format and syntax descriptions.

**<>** 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.

... An ellipsis indicates that you can repeat the preceding item one or more times.

**<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>`.
What’s new

This section summarizes all the changes made to IBM General Parallel File System for AIX 5L:

What’s new for GPFS 2.1

GPFS 2.1 provides several usability enhancements:

• Support for AIX 5L 5.1 with APAR IY33002 including:
  – The ability to create a GPFS cluster from an RSCT peer domain.
  – Faster failover through the persistent reserve feature.

• Support the latest IBM @server Cluster 1600 configuration

• The GPFS for AIX 5L product may be installed in either an AIX cluster environment or a PSSP cluster environment. Consequently, two sets of man pages are now shipped with the product and you must set your MANPATH environment variable accordingly (see Installing the GPFS manual pages).

• 64-bit kernel exploitation
  The GPFS kernel extensions are now shipped in both 32-bit and 64-bit formats.

• Electronic license agreement

• Two new commands for managing the disks (logical volumes) in your GPFS cluster:
  – mmdellv
  – mmlsgpfsdisk

• For atime and mtime values as reported by the stat, fstat, gpfs_stat and gpfs_fstat calls, you may:
  – Suppress updating the value of atime.
    When suppressing the periodic update, these calls will report the time the file was last accessed when the file system was mounted with the -S no option or, for a new file, the time the file system was created.
  – Display the exact value for mtime.
    The default is to periodically update the mtime value for a file system. If it is more desirable to display exact modification times for a file system, specify the -E yes option.

Commands which have been updated:

1. mmcrfs
2. mmchfs
3. mmlsfs

• The capability to read from or write to a file with direct I/O. The mmchattr command has been updated with the -D option for this support.

• The default use designation for nodes in your GPFS nodeset has been changed from manager to client.

Commands which have been updated:

1. mmconfig
2. mmchconfig

• The terms to install/uninstall GPFS quotas have been replaced by the terms enable/disable GPFS quota management.

• The GPFS documentation is no longer shipped on the product CD-ROM. You may download, view, search, and print the supporting documentation for the GPFS program product in the following ways:
  1. In PDF format:
     – From the IBM Publications Center at www.ibm.com/shop/publications/order
2. In HTML format at publib.boulder.ibm.com/clresctr/docs/gpfs/html

To view the GPFS PDF publications, you need access to Adobe Acrobat Reader. Acrobat Reader is shipped with the AIX 5L Bonus Pack and is also freely available for downloading from the Adobe website at www.adobe.com. Since the GPFS documentation contains cross-book links, if you choose to download the PDF files they should all be placed in the same directory and the files should not be renamed.

To view the GPFS HTML publications, you need access to an HTML document browser such as Netscape. An index file into the HTML files (aix_index.html) is provided when downloading the tar file of the GPFS HTML publications. Since the GPFS documentation contains cross-book links, all files contained in the tar file should remain in the same directory.

The GPFS library includes:

– General Parallel File System for AIX 5L: AIX Clusters Administration and Programming Reference, SA22-7896 (PDF file name an2adm00.pdf)
– General Parallel File System for AIX 5L: AIX Clusters Problem Determination Guide, GA22-7897 (PDF file name an2pdg00.pdf)
– General Parallel File System for AIX 5L: AIX Clusters Data Management API Guide, GA22-7898 (PDF file name an2dmp00.pdf)

New file system functions existing in GPFS 2.1 are not usable in existing file systems until you explicitly authorize these changes by issuing the `mmchfs -V` command.

Migration

For information on migrating your system to the latest level of GPFS, see Migration, coexistence, and compatibility.
Part 1. Understanding GPFS

Part 1 provides planning concepts for the General Parallel File System for AIX 5L (GPFS) licensed program:

- Chapter 1, "Introducing General Parallel File System", on page 3
- Chapter 2, "Planning for GPFS", on page 9
Chapter 1. Introducing General Parallel File System

IBM's General Parallel File System (GPFS) allows users shared access to files that may span multiple disk drives on multiple nodes. It offers many of the standard UNIX® file system interfaces allowing most applications to execute without modification or recompiling. UNIX file system utilities are also supported by GPFS. That is, users can continue to use the UNIX commands they have always used for ordinary file operations (see Appendix B, "Considerations for GPFS applications", on page 59 for exceptions). The only unique commands are those for administering the GPFS file system (see the General Parallel File System for AIX 5L: AIX Clusters Administration and Programming Reference for complete command usage information).

GPFS provides file system services to parallel and serial applications. GPFS allows parallel applications simultaneous access to the same files, or different files, from any node in the GPFS nodeset while managing a high level of control over all file system operations. A nodeset is a group of nodes that all run the same level of GPFS and operate on the same file system.

GPFS is particularly appropriate in an environment where the aggregate peak need for data exceeds the capability of a distributed file system server. It is not appropriate for those environments where hot backup is the main requirement or where data is readily partitioned along individual node boundaries.

The strengths of GPFS

GPFS is a powerful file system offering:

- Improved system performance
- Assured file consistency
- High recoverability and increased data availability
- Enhanced system flexibility
- Simplified administration

Improved system performance

Using GPFS to store and retrieve your files can improve system performance by:

- Allowing multiple processes or applications on all nodes in the nodeset simultaneous access to the same file using standard file system calls.
- Increasing aggregate bandwidth of your file system by spreading reads and writes across multiple disks.
- Balancing the load evenly across all disks to maximize their combined throughput. One disk is no more active than another.
- Supporting large amounts of data.
- Allowing concurrent reads and writes from multiple nodes. This is a key concept in parallel processing.

Assured file consistency

GPFS uses a sophisticated token management system to provide data consistency while allowing multiple independent paths to the same file by the same name from anywhere in the system. Even when nodes are down or hardware resource demands are high, GPFS can find an available path to file system data.

High recoverability and increased data availability

GPFS is a logging file system that creates separate logs for each node. These logs record the allocation and modification of metadata aiding in fast recovery and the restoration of data consistency in the event of node failure.

GPFS failover support allows you to organize your hardware into a number of failure groups to minimize single points of failure. A failure group is a set of disks that share a common point of failure that could
cause them all to become simultaneously unavailable. In order to assure file availability, GPFS maintains each instance of replicated data on disks in different failure groups.

The replication feature of GPFS allows you to determine how many copies of a file to maintain. File system replication assures that the latest updates to critical data are preserved in the event of disk failure. During configuration, you assign a replication factor to indicate the total number of copies you wish to store. Replication allows you to set different levels of protection for each file or one level for an entire file system. Since replication uses additional disk space and requires extra write time, you might want to consider replicating only file systems that are frequently read from but seldom written to (see “File system recoverability parameters” on page 22). Even if you do not specify replication when creating a file system, GPFS automatically replicates recovery logs in separate failure groups. For further information on failure groups see “Logical volume creation considerations” on page 11.

Once your file system is created, you can have it automatically mounted whenever the GPFS daemon is started. The automount feature assures that whenever the system and disks are up, the file system will be available.

**Enhanced system flexibility**

With GPFS, your system resources are not frozen. You can add or delete disks while the file system is mounted. When the time is right and system demand is low, you can rebalance the file system across all currently configured disks. You can also add new nodes without having to stop and restart the GPFS daemon (an exception to this applies when single-node quorum is in effect, see “Quorum” on page 18).

After GPFS has been configured for your system, depending on your applications, hardware, and workload, you can reconfigure GPFS to increase throughput. Set up your GPFS environment for today's applications and users, secure in the knowledge that you can expand in the future without jeopardizing your data. GPFS capacity can grow as your hardware expands.

**Simplified administration**

GPFS commands save configuration and file system information in one or more files, collectively known as **GPFS cluster data**. The GPFS administration commands are designed to keep these files synchronized between each other and with the GPFS system files on each node in the nodeset, thereby ensuring accurate configuration data (see “GPFS cluster data” on page 57).

GPFS administration commands are similar in name and function to UNIX file system commands, with one important difference: *the GPFS commands operate on multiple nodes*. A single GPFS command performs a file system function across the entire nodeset. Most GPFS administration tasks can be performed from any node running GPFS (see the individual commands as documented in the **General Parallel File System for AIX 5L: AIX Clusters Administration and Programming Reference**).
The basic GPFS structure

GPFS is a clustered file system defined over a number of nodes. The overall set of nodes over which GPFS is defined is known as a GPFS cluster. Depending on the operating environment, GPFS defines several cluster types:

Table 1. GPFS cluster types

<table>
<thead>
<tr>
<th>Cluster type</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp</td>
<td>The PSSP cluster environment is based on the IBM Parallel System Support Programs (PSSP) program product and the shared disk concept of the IBM Virtual Shared Disk program product. In the PSSP cluster environment, the boundaries of the GPFS cluster depend on the switch type being used. In a system with an SP Switch, the GPFS cluster is equal to the corresponding SP partition. In a system with an SP Switch2, the GPFS cluster is equal to all of the nodes in the system. For information regarding the GPFS for AIX 5L licensed program for PSSP clusters go to <a href="http://www.ibm.com/servers/eserver/pseries/software/sp/gpfs.html">www.ibm.com/servers/eserver/pseries/software/sp/gpfs.html</a>.</td>
</tr>
</tbody>
</table>
| rpd or hacmp | The AIX cluster environment is based on either:  
- A Reliable Scalable Cluster Technology (RSCT) peer domain created by the RSCT subsystem of AIX 5L. With an RSCT peer domain, all nodes in the GPFS cluster have the same view of the domain and share the resources within the domain (GPFS cluster type rpd).  
- An HACMP cluster created by the High Availability Cluster Multi-Processing for AIX/Enhanced Scalability (HACMP/ES) program product (GPFS cluster type hacmp). In the AIX cluster environment, the boundaries of the GPFS cluster are maintained with the mmcrcluster, mmaddcluster, and mmdelcluster commands. |
| lc           | The loose cluster environment is based on the Linux® operating system. In a loose cluster environment, the boundaries of the GPFS cluster are maintained with the mmcrcluster, mmaddcluster, and mmdelcluster commands. For information regarding the GPFS for Linux licensed program go to [www.ibm.com/servers/eserver/clusters/software/gpfs.html](http://www.ibm.com/servers/eserver/clusters/software/gpfs.html). |

Within a GPFS cluster, the nodes are divided into one or more GPFS nodesets. The nodes in each nodeset share a set of file systems which are not accessible by the nodes in any other nodeset.

On each node in the cluster, GPFS consists of:
1. Administration commands
2. A kernel extension
3. A multi-threaded daemon

For a detailed discussion of GPFS, see Appendix A, “GPFS architecture”, on page 45.

The GPFS kernel extension

The GPFS kernel extension provides the interfaces to the operating system VNODE and virtual file system (VFS) interfaces for adding a file system. GPFS kernel extensions exist in both 32-bit and 64-bit forms. See Compatibility. Structurally, applications make file system calls to the operating system, which presents them to the GPFS file system kernel extension. In this way, GPFS appears to applications as just another file system. The GPFS kernel extension will either satisfy these requests using resources which are already available in the system, or send a message to the GPFS daemon to complete the request.
The GPFS daemon

The GPFS daemon performs all I/O and buffer management for GPFS. This includes read-ahead for sequential reads and write-behind for all writes not specified as synchronous. All I/O is protected by token management, which ensures that the file system on multiple nodes honors the atomicity and provides data consistency of a file system.

The daemon is a multi-threaded process with some threads dedicated to specific functions. This ensures that services requiring priority attention are not blocked because other threads are busy with routine work. The daemon also communicates with instances of the daemon on other nodes to coordinate configuration changes, recovery and parallel updates of the same data structures. Specific functions that execute on the daemon include:

1. Allocation of disk space to new files and newly extended files. This is done in coordination with the file system manager (see "The file system manager" on page 45).
2. Management of directories including creation of new directories, insertion and removal of entries into existing directories, and searching of directories that require I/O.
3. Allocation of appropriate locks to protect the integrity of data and metadata. Locks affecting data that may be accessed from multiple nodes require interaction with the token management function.
4. Disk I/O is initiated on threads of the daemon.
5. Security and quotas are also managed by the daemon in conjunction with the file system manager.

The AIX cluster environment

In an AIX cluster environment, GPFS is designed to operate with:

AIX 5L

providing:

- The basic operating system services and the routing of file system calls requiring GPFS data.
- The LVM subsystem for direct disk management.
- Persistent reserve for transparent failover of disk access in the event of disk failure.

and either the

Reliable Scalable Cluster Technology (RSCT) subsystem of AIX 5L

providing the capability to create, modify, and manage an RSCT peer domain:

- The Resource Monitoring and Control (RMC) component establishing the basic cluster environment, monitoring the changes within the domain, and enabling resource sharing within the domain.
- The Group Services component coordinating and synchronizing the changes across nodes in the domain thereby maintaining the consistency in the domain.
- The Topology Services component providing network adapter status, node connectivity, and a reliable messaging service.
- The configuration manager employs the above subsystems to create, change, and manage the RSCT peer domain.

or the

HACMP/ES program product

providing:

- The basic cluster operating environment.
- The Group Services component coordinating and synchronizing the changes across nodes in the HACMP cluster thereby maintaining the consistency in the cluster.
- The Topology Services component providing network adapter status, node connectivity, and a reliable messaging service.
The RSCT peer domain environment

In an RSCT peer domain environment, a GPFS cluster is a group of RS/6000 machines, @server pSeries machines, or a mixture of both with uniform disk access enabling concurrent data sharing. The GPFS cluster is created from an existing RSCT peer domain. There can only be one GPFS cluster per RSCT peer domain. Within that GPFS cluster, you may define multiple GPFS nodesets. However, a node may only belong to one nodeset. For further information on the RSCT component of AIX 5L and the associated subsystems, see the *Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*.

In this environment, the size of your GPFS nodeset is constrained by the type of disk attachment. If any of the disks in the file system are SSA disks, your nodeset may consist of up to eight RS/6000 or @server pSeries machines (the size of the nodeset is constrained by the limitations of the SSA adapter). If the disks in the file system are purely Fibre Channel, your nodeset may consist of up to 32 RS/6000 or @server pSeries machines (the size of the nodeset is constrained by the limitations of the Group Services software). When a GPFS nodeset is being configured, or nodes are being added to or deleted from the cluster, GPFS obtains the necessary additional configuration data from the resource classes maintained by the RSCT peer domain:

1. node number (PeerNode resource class)
2. adapter type (NetworkInterface resource class)
3. IP address (NetworkInterface resource class)

The complete configuration data maintained by GPFS is then stored on the primary, and if specified, the secondary GPFS cluster data server as designated on the `mmcrcluster` command (see "GPFS cluster creation considerations" on page 14).

![Diagram of an RSCT peer domain environment](image)

*Figure 1. An RSCT peer domain environment*

For complete hardware and programming specifications, see "Hardware specifications" on page 9 and "Programming specifications" on page 9.

The HACMP environment

In the HACMP environment, a GPFS cluster is a group of RS/6000 machines, @server pSeries machines, or a mixture of both with uniform disk access enabling concurrent data sharing. The GPFS cluster is created from an existing HACMP cluster. There can only be one GPFS cluster per HACMP cluster. Within that GPFS cluster, you may define multiple GPFS nodesets. However, a node may only belong to one nodeset. For further on the HACMP/ES program product, see the *High Availability Cluster Multi-Processing for AIX: Enhanced Scalability Installation and Administration Guide*.

In this environment, the size of your GPFS nodeset is constrained by the type of disk attachment. If any of the disks in the file system are SSA disks, your nodeset may consist of up to eight RS/6000 or @server
pSeries machines (the size of the nodeset is constrained by the limitations of the SSA adapter). If the disks in the file system are purely Fibre Channel, your nodeset may consist of up to 32 RS/6000 or @server pSeries machines (the size of the nodeset is constrained by the limitations of the HACMP/ES software). After a GPFS nodeset has been configured, or nodes have been added to or deleted from the nodeset, GPFS obtains the necessary additional configuration data from the HACMP/ES Global Object Data Manager (ODM):
1. node number
2. adapter type
3. IP address

The complete configuration data maintained by GPFS is then stored on the primary, and if specified, the secondary GPFS cluster data server as designated on the `mmcrcluster` command (see “GPFS cluster creation considerations” on page 14).

For complete hardware and programming specifications, see “Hardware specifications” on page 9 and “Programming specifications” on page 9.
Chapter 2. Planning for GPFS

Planning for GPFS includes:
- "Hardware specifications"
- "Programming specifications"
- "Recoverability considerations"
- "Disk considerations" on page 11
- "GPFS cluster creation considerations" on page 14
- "Nodeset configuration considerations" on page 15
- "File system creation considerations" on page 20

Although you can modify your GPFS configuration after it has been set, a little consideration before installation and initial setup will reward you with a more efficient and immediately useful file system. During configuration, GPFS requires you to specify several operational parameters that reflect your hardware resources and operating environment. During file system creation, you have the opportunity to specify parameters based on the expected size of the files or allow the default values to take effect. These parameters define the disks for the file system and how data will be written to them.

Hardware specifications
1. An existing IBM Eserver configuration:
   - An RSCT peer domain established with the RSCT component of AIX 5L
     For information on creating an RSCT peer domain, see the [Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference](#).
   - An HACMP cluster established with the HACMP/ES program product
     For information on creating an HACMP cluster, see the [High Availability Cluster Multi-Processing for AIX: Enhanced Scalability Installation and Administration Guide](#).
2. Enough disks to contain the file system (see "Disk considerations" on page 11).
3. An IP network of sufficient network bandwidth (minimum of 100Mb per second).

Programming specifications
1. AIX 5L Version 5 Release 1 (5765-E61) with IY30258, or later modifications
2. For a GPFS cluster type hacmp, HACMP/ES version 4.4.1 (5765-E54), or later modifications

Recoverability considerations
Sound file system planning includes considering replication as well as structuring your data so information is not vulnerable to a single point of failure. GPFS provides you with parameters that enable you to create a highly available file system with fast recoverability from failures. At the file system level, the metadata and data replication parameters are set (see "File system recoverability parameters" on page 22). At the disk level when preparing disks for use with your file system, you can specify disk usage and failure group positional parameters to be associated with each disk (see "Logical volume creation considerations" on page 11).

Additionally, GPFS provides several layers of protection against failures of various types:
1. "Node failure" on page 10
2. "Disk failure" on page 10
3. "Making your decision" on page 10
Node failure
This basic layer of protection covers the failure of file system nodes and is provided by Group Services. When an inoperative node is detected by Group Services, GPFS fences it out using environment-specific subsystems (see [Disk fencing](#)). This prevents any write operations that might interfere with recovery.

File system recovery from node failure should not be noticeable to applications running on other nodes, except for delays in accessing objects being modified on the failing node. Recovery involves rebuilding metadata structures, which may have been under modification at the time of the failure. If the failing node is the file system manager for the file system, the delay will be longer and proportional to the activity on the file system at the time of failure, but no administrative intervention will be needed.

During node failure situations, if multi-node quorum is in effect, quorum needs to be maintained in order to recover the failing nodes. If multi-node quorum is not maintained due to node failure, GPFS restarts on all nodes, handles recovery, and attempts to achieve quorum again.

Disk failure
The most common reason why data becomes unavailable is disk failure with no redundancy. In the event of disk failure, GPFS discontinues use of the disk and awaits its return to an available state. You can guard against loss of data availability from such failures by setting the GPFS recoverability parameters (replication, disk usage, and failure group designations) either alone or in conjunction with one of these environment specific methods to maintain additional copies of files.

One means of data protection is the use of a RAID/Enterprise Storage Subsystem (ESS) controller, which masks disk failures with parity disks. An ideal configuration is shown in Figure 3, where a RAID/ESS controller is multi-tailed to each node in the nodeset.

![Figure 3. RAID/ESS Controller multi-tailed to each node](#)

Making your decision
Each method of data protection has its cost, whether it be the installation of additional hardware or the consumption of large amounts of disk space. If your configuration consists of:
- SSA disks and you have greater than two nodes in the GPFS cluster, GPFS replication is the only data protection available to you.
- SSA disks with either one or two nodes, you can use both SSA RAID and GPFS replication.
- Fibre Channel disks with any number of nodes, you can use both RAID and GPFS replication.
Disk considerations

You may have up to 1024 external shared disks or disk arrays with the adapters configured to allow each disk connectivity to each node in the nodeset. No disk can be larger than 1 TB.

Proper planning for your disk subsystem includes determining:

- Sufficient disks to meet the expected I/O load
- Sufficient connectivity (adapters and buses) between disks

Disks can be attached using:

- SSA
- Fibre Channel
- Enterprise Storage Server™ (ESS) in either Subsystem Device Driver (SDD) or non-SDD mode

The actual number of disks in your system may be constrained by products other than GPFS which you have installed. Refer to individual product documentation for support information.

Disk considerations include:

- "Disk fencing"
- "Logical volume creation considerations"

Disk fencing

In order to preserve data integrity in the event of certain system failures, GPFS will fence a node that is down from the file system until it returns to the available state. Depending upon the types of disk you are using, there are three possible ways for the fencing to occur:

**SSA fencing**

SSA disks

**SCSI-3 persistent reserve**

For a list of GPFS supported persistent reserve devices, see the Frequently Asked Questions at [www.ibm.com/servers/eserver/clusters/library/](http://www.ibm.com/servers/eserver/clusters/library/)

**disk leasing**

A GPFS specific fencing mechanism for disks which do not support either SSA fencing or SCSI-3 persistent reserve.

Single-node quorum is only supported when disk leasing is not in effect. Disk leasing is activated if any disk in any file system in the nodeset is not using SSA fencing or SCSI-3 persistent reserve.

Logical volume creation considerations

You must prepare each physical disk you intend to use with GPFS as a logical volume. This is done via the `mmcrlv` command. Disks are identified to the `mmcrlv` command by their physical disk device names.

Notes:

1. A PVID must exist on each disk being used by GPFS on each node in the cluster prior to issuing the `mmcrlv` command. If a valid PVID does not exist, the command will fail upon importing the logical volume:
   a. Verify the existence of a PVID by issuing the `lspv` command. The system displays information similar to:
      
      ```
      lspv
      hdisk3  0020570a72bba1a0  None
      hdisk4  none  None
      ```
   b. If a PVID does not exist, prior to assigning a PVID you must ensure that the disk is not a member of a mounted and active GPFS file system. If the disk is a member of an active and mounted
GPFS file system and you issue the `chdev` command to assign a PVID, there is the possibility you will experience I/O problems which may result in the file system being unmounted on one or more nodes.

c. To assign a PVID, issue the `chdev` command:

```bash
chdev -l hdisk4 -a pv=yes
```

The system displays information similar to:

```
hdisk4 changed
```

To determine the PVID assign, issue the `lspv` command. The system displays information similar to:

```
lspv
hdisk3 0020570a72bbb1a0 None
hdisk4 0022b60ade92fb24 None
```

2. Single-node quorum is only supported when disk leasing is not in effect. Disk leasing is activated if any disk in any file system in the nodeset is not using SSA fencing or SCSI-3 persistent reserve.

3. Logical volumes created by the `mmcrlv` command will:
   - Use SCSI-3 persistent reserve on disks which support it or SSA fencing if that is supported by the disk. Otherwise disk leasing will be used.
   - Have bad-block relocation automatically turned off. Accessing disks concurrently from multiple systems using lvm bad-block relocations could potentially cause conflicting assignments. As a result, software bad-block relocation is turned off allowing the hardware bad-block relocation supplied by your disk vendor to provide protection against disk media errors.

4. You cannot protect your file system against disk failure by mirroring data at the LVM level. You must use GPFS replication or RAID devices to protect your data (see “Recoverability considerations” on page 9).

5. In an HACMP environment, any disk resources (volume groups and logical volumes) that will be used by GPFS must not belong to any HACMP/ES resource group. HACMP/ES will not be in control of these disk resources and is not responsible for varying them on or off at any time. The responsibility to keep the disks in the proper state belongs to GPFS in the HACMP environment. For further information on logical volume concepts, see the AIX 5L System Management Guide: Operating System and Devices.

The `mmcrlv` command expects as input a file, `DescFile`, containing a disk descriptor, one per line, for each of the disks to be processed. Disk descriptors have the format (second and third fields reserved):

```
DiskName:::DiskUsage:FailureGroup
```

**DiskName**

The physical device name of the disk you want to define as a logical volume. This is the `/dev` name for the disk on the node on which the `mmcrlv` command is issued and can be either an hdisk name or a vpath name for an SDD device. Each disk will be used to create a single volume group and a single logical volume.

**Disk Usage**

What is to be stored on the disk. `metadataOnly` specifies that this disk may only be used for metadata, not for data. `dataOnly` specifies that only data, and not metadata, is allowed on this disk. You can limit vulnerability to disk failure by confining metadata to a small number of conventional mirrored or replicated disks. The default, `dataAndMetadata`, allows both on the disk.

**Note:** RAID devices are not well-suited for performing small block writes. Since GPFS metadata writes are often smaller than a full block, you may find using non-RAID devices for GPFS metadata better for performance.
**FailureGroup**

A number identifying the failure group to which this disk belongs. All disks that are either attached to the same adapter have a common point of failure and should therefore be placed in the same failure group.

GPFS uses this information during data and metadata placement to assure that no two replicas of the same block will become unavailable due to a single failure. You can specify any value from -1 (where -1 indicates that the disk has no point of failure in common with any other disk) to 4000. If you specify no failure group, the value defaults to -1.

Upon successful completion of the `mmcrlv` command, these tasks are completed on all nodes in the GPFS cluster:

- For each valid descriptor in the descriptor file, local logical volumes and the local volume groups are created.

  The logical volume names are assigned according to the convention:

  `gpfsNNlv`

  where `NN` is a unique non-negative integer not used in any prior logical volume named with this convention.

  The local volume group component of the logical volume is named according to the same convention:

  `gpfsNNvg`

  where `NN` is a unique non-negative integer not used in any prior local volume group named with this convention.

- The physical device or vpath name is replaced with the created logical volume names.
- The local volume groups are imported to all available nodes in the GPFS cluster.
- The `DescFile` is rewritten to contain the created logical volume names in place of the physical disk or vpath name and all other fields, if specified, are copied without modification. The rewritten disk descriptor file can then be used as input to the `mmcrfs`, `mmadddisk`, or `mmrpldisk` commands. If you do not use this file, you must accept the default values or specify these values when creating disk descriptors for subsequent `mmcrfs`, `mmadddisk`, or `mmrpldisk` commands.

If necessary, the `DiskUsage` and `FailureGroup` values for a disk can be changed with the `mmchdisk` command.
GPFS cluster creation considerations

A GPFS cluster is created by issuing the `mmcrcluster` command. Table 2 details the GPFS cluster creation options on the `mmcrcluster` command, which options can be changed later by the `mmchcluster` command, and what the default values are.

Table 2. GPFS cluster creation options

<table>
<thead>
<tr>
<th>mmcrcluster</th>
<th>mmchcluster</th>
<th>default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes in your GPFS cluster</td>
<td>X</td>
<td>To add or delete nodes from the cluster use <code>mmaddcluster</code> or <code>mmdelcluster</code> respectively</td>
</tr>
<tr>
<td>GPFS cluster data servers primary</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GPFS cluster data servers secondary</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GPFS cluster type on page 15</td>
<td>X</td>
<td>This cannot be changed</td>
</tr>
<tr>
<td>Remote shell command on page 15</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Remote file copy command on page 15</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Notes:
1. X – indicates the option is available on the command
2. an empty cell – indicates the option in not available on the command

Nodes in your GPFS cluster

When you create your GPFS cluster you must provide a file containing a list of nodes to be included in the cluster. During creation of your cluster, GPFS copies this information to the GPFS cluster data server.

The file lists one node per line. The hostname or IP address used for a node must refer to the adapter port over which the GPFS daemons communicate. Alias interfaces are not allowed. Use the original address or a name that is resolved by the `host` command to that original address. You may specify a node using any of these forms:

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short hostname</td>
<td>k145n01</td>
</tr>
<tr>
<td>Long hostname</td>
<td>k145n01.kgn.ibm.com</td>
</tr>
<tr>
<td>IP address</td>
<td>9.119.19.102</td>
</tr>
</tbody>
</table>

You must follow these rules when creating your GPFS cluster:
- A node may only belong to one GPFS cluster at a time.
- The node must be a properly configured member of either your RSCT peer domain or your HACMP cluster.
- The node must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node, create a new input file containing the failed nodes only, and reissue the `mmaddcluster` command to add those nodes.

GPFS cluster data servers

From the nodes included in your GPFS cluster, you **must** designate one of the nodes as the primary GPFS cluster data server on which GPFS configuration information is maintained. It is suggested that you also specify a secondary GPFS cluster data server. If your primary server fails and you have not designated a...
backup server, the GPFS cluster data is inaccessible and any GPFS administrative command that is
issued will fail. Similarly, when the GPFS daemon starts up, at least one of the two GPFS cluster data
server nodes must be accessible (see “GPFS cluster data” on page 57).

**GPFS cluster type**
The only valid GPFS cluster types are either rpd or hacmp. Specifying any other cluster type will cause the
mmcrcluster command to fail.

**Remote shell command**
The default remote shell command is rsh. This requires that a properly configured .rhosts file exist in the
root user’s home directory on each node in the GPFS cluster.

If you choose to designate the use of different remote shell command on either the mmcrcluster or the
mmchcluster command, you must specify the fully qualified pathname for the program to be used by
GPFS. You must also ensure:
1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password.

The remote shell command must adhere to the same syntax form as rsh but may implement an alternate
authentication mechanism.

**Remote file copy command**
The default remote file copy program is rcp. This requires that a properly configured .rhosts file exist in the
root user’s home directory on each node in the GPFS cluster.

If you choose to designate the use of different remote file copy command on either the mmcrcluster or the
mmchcluster command, you must specify the fully qualified pathname for the program to be used by
GPFS. You must also ensure:
1. Proper authorization is granted to all nodes in the GPFS cluster.
2. The nodes in the GPFS cluster can communicate without the use of a password.

The remote copy command must adhere to the same syntax form rcp but may implement an alternate
authentication mechanism.

**Nodeset configuration considerations**
Before you configure your GPFS nodeset:
1. ensure you have tuned your system (see Chapter 4, “Tuning your system for GPFS”, on page 35).
2. You must first create a GPFS cluster (see “GPFS cluster creation considerations” on page 14).

Configuration involves defining the nodes to be included in the GPFS nodeset and specifying how they will
operate. Your GPFS nodeset is configured by issuing the mmconfig command. Table 3 details the
configuration options specified on the mmconfig command, which options can be changed later with the
mmchconfig command, and what the default values are.

<table>
<thead>
<tr>
<th>Table 3. GPFS configuration options</th>
</tr>
</thead>
<tbody>
<tr>
<td>mmconfig</td>
</tr>
<tr>
<td>&quot;Nodes in your GPFS nodeset&quot; on page 16</td>
</tr>
<tr>
<td>Table 3. GPFS configuration options (continued)</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>&quot;Nodeset identifier&quot; on page 17</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>&quot;Starting GPFS automatically&quot; on page 17</td>
</tr>
<tr>
<td>&quot;Path for the storage of dumps&quot; on page 17</td>
</tr>
<tr>
<td>&quot;Quorum&quot; on page 18</td>
</tr>
<tr>
<td>&quot;pagepool&quot; on page 18</td>
</tr>
<tr>
<td>&quot;maxFilesToCache&quot; on page 18</td>
</tr>
<tr>
<td>&quot;maxStatCache&quot; on page 18</td>
</tr>
<tr>
<td>&quot;Maximum file system block size allowed&quot; on page 19</td>
</tr>
<tr>
<td>&quot;dmapiEventTimeout&quot; on page 19</td>
</tr>
<tr>
<td>&quot;dmapiSessionFailureTimeout&quot; on page 19</td>
</tr>
<tr>
<td>&quot;dmapiMountTimeout&quot; on page 19</td>
</tr>
</tbody>
</table>

Notes:
1. X – indicates the option is available on the command
2. an empty cell – indicates the option is not available on the command

Nodes in your GPFS nodeset

You can provide a list of nodes as input to the `mmconfig` command or allow GPFS to configure all of the nodes in the GPFS cluster. If the disks in your nodeset are SSA or a combination of SSA and Fibre Channel, the maximum number of nodes in the nodeset is eight. If your disks are purely Fibre Channel, the maximum number of nodes in a nodeset is 32.

If a node is down or is not a member of the GPFS cluster, the `mmconfig` command fails. If the node is down when the `mmconfig` command is issued, when the nodes comes back up, add it to the nodeset by issuing the `mmaddnode` command. If the node is not a member of the GPFS cluster (see "GPFS cluster creation considerations" on page 14), you must:
1. Issue the `mmaddcluster` command.
2. Re-issue the `mmconfig` command.

Within the GPFS cluster, you may define multiple GPFS nodesets. However, a node may only belong to one nodeset. After a GPFS nodeset has been configured, or nodes have been added to or deleted from the nodeset, the information is maintained on the GPFS cluster data server.

When specifying a list of nodes, the name of this list must be specified with the `-n` option on the `mmconfig` command. The list must contain only one entry per line. Nodes are specified by a `NodeName` and may be optionally followed by a use designation:

Node desctriptors have the format:

*NodeName*: `manager | client`

*NodeName*

The hostname or IP address used for a node must refer to the communications adapter over
which the GPFS daemons communicate. Alias interfaces are not allowed. Use the original address
or a name that is resolved by the host command to that original address.

You may specify a node using any of these forms:

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short hostname</td>
<td>k145n01</td>
</tr>
<tr>
<td>Long hostname</td>
<td>k145n01.kgn.ibm.com</td>
</tr>
<tr>
<td>IP address</td>
<td>9.119.19.102</td>
</tr>
</tbody>
</table>

manager | client
An optional use designation.

The designation specifies whether or not the node should be included in the pool of nodes from
which the file system manager is chosen (the special functions of the file system manager
consume extra processing time see “The file system manager” on page 45). The default is to not
have the node included in the pool.

In general, small systems (less than 128 nodes) do not need multiple nodes dedicated for the file
system manager. However, if you are running large parallel jobs, threads scheduled to a node
performing these functions may run slower. As a guide, in a large system there should be one file
system manager node for each GPFS file system.

Nodeset identifier
You can provide a name for the nodeset by using the -C option on the mmconfig command or allow
GPFS to assign one. If you choose the identifier, it can be at most eight alphanumeric characters long and
may not be a reserved word or the number zero. If GPFS assigns one, it will be an integer identifier
beginning with the value one and increasing sequentially as nodesets are added. This designation may not
be changed once it is assigned.

The operation of nodes
In deciding how your nodeset will operate, you must consider:

- Starting GPFS automatically
- Path for the storage of dumps
- “Quorum” on page 18
- “Cache usage” on page 18
- “Maximum file system block size allowed” on page 19
- “DMAP configuration options” on page 19

Starting GPFS automatically
You can configure GPFS to start automatically on all nodes in the nodeset whenever they come up, by
specifying the autoload (-A) option for the mmconfig command. This eliminates the need to start GPFS by
issuing the mmstartup command. The default is not to start the daemon automatically.

Path for the storage of dumps
The default is to store dumps in /tmp/mmfs, however you may specify an alternate path. You may also
specify no if you do not want to store any dumps.

It is suggested that you create a directory for the storage of dumps as this will contain certain problem
determination information. This can be a symbolic link to another location if more space can be found
there. It should not be placed in a GPFS file system as it might not be available should GPFS fail. If a
problem should occur, GPFS may write 200 MB or more of problem determination data into the directory.
These files must be manually removed when any problem determination is complete. This should be done
promptly so that a no space condition is not encountered if another failure occurs.
**Quorum**

For all nodesets consisting of three or more nodes, GPFS quorum is defined as one plus half of the number of nodes in the GPFS nodeset (referred to as multi-node quorum). For a two-node nodeset, you have the choice of allowing multi-node quorum or specifying the `-U` option on the `mmconfig` command to indicate the use of a single-node quorum. The specification of single-node quorum allows the remaining node in a two-node nodeset to continue functioning in the event of the failure of the peer node.

**Note:** Single-node quorum is only supported when disk leasing is not in effect. Disk leasing is activated if any disk in any file system in the nodeset is not using SSA fencing or SCSI-3 persistent reserve.

If multi-node quorum is used, quorum needs to be maintained in order to recover the failing nodes. If multi-node quorum is not maintained due to node failure, all GPFS nodes restart, handle recovery, and attempt to achieve quorum again. Therefore, in a three-node system, failure of one node will allow recovery and continued operation on the two remaining nodes. This is the minimum configuration where continued operation is possible due to the failure of a node. That is, in a two-node system where single-node quorum has not been specified, the failure of one node means both nodes will restart, handle recovery, and attempt to achieve quorum again.

If single-node quorum is specified, the failure of one node results in GPFS fencing the failing node from the disks containing GPFS file system data. The remaining node will continue processing if the fencing operation was successful. If not, those file systems which could not be completely fenced will be unmounted and attempts to fence the node will continue (in the unlikely event that both nodes end up fenced, see the [General Parallel File System for AIX 5L: AIX Clusters Problem Determination Guide](https://www.ibm.com/support/knowledgecenter/SSLTBW_AIX_51/ls/ls_cluster/mem_overview.html#single-nodemem) and search on `single-node quorum`).

**Cache usage**

GPFS creates a number of cache segments on each node in the nodeset. The amount of cache is controlled by three parameters:

**pagepool**

The amount of pinned memory reserved for caching data read from disk. This consists mainly of file data, but also includes directory blocks and other file system metadata such as indirect blocks and allocation maps (see Appendix A, “GPFS architecture”, on page 45). `pagepool` is used for read-ahead and write-behind operations to increase performance, as well as for reuse of cached data.

The size of the cache on each node can range from a minimum of 4 MB to a maximum of 512 MB. For systems where applications access large files, reuse data, or have a random I/O pattern, increasing the value for `pagepool` may prove beneficial. This value must be specified with the character `M`, for example `80M`. The default is `20M`.

**maxFilesToCache**

The total number of different files that can be cached at one time. Every entry in the file cache requires some pageable memory to hold the content of the file’s inode plus control data structures. This is in addition to any of the file’s data and indirect blocks that might be cached in the page pool.

The total amount of memory required for inodes and control data structures can be calculated as:

\[
\text{maxFilesToCache} \times 2.5 \text{ KB}
\]

where \(2.5 \text{ KB} = 2 \text{ KB} + 512 \text{ bytes for an inode}\)

Valid values of `maxFilesToCache` range from 0 to 1,000,000. For systems where applications use a large number of files, of any size, increasing the value for `maxFilesToCache` may prove beneficial (this is particularly true for systems where a large number of small files are accessed). The value should be large enough to handle the number of concurrently open files plus allow caching of recently used files. The default value is 1000.
maxStatCache
This parameter sets aside additional pageable memory to cache attributes of files that are not currently in the regular file cache. This is useful to improve the performance of both the system and GPFS stat() calls for applications with a working set that does not fit in the regular file cache.

The memory occupied by the stat cache can be calculated as:

\[ \text{maxStatCache} \times 176 \text{ bytes} \]

Valid values of maxStatCache range from 0 to 1,000,000. For systems where applications test the existence of files, or the properties of files, without actually opening them (as backup applications do), increasing the value for maxStatCache may prove beneficial. The default value is:

\[ 4 \times \text{maxFilesToCache} \]

The total amount of memory GPFS uses to cache file data and metadata is arrived at by adding pagepool to the amount of memory required to hold inodes and control data structures (maxFilesToCache \times 2.5 KB), and the memory for the stat cache (maxStatCache \times 176 bytes) together. The combined amount of memory to hold inodes, control data structures, and the stat cache is limited to 50% of the physical memory. With an inode size of 512 bytes, the default 4-to-1 ratio of maxStatCache to maxFilesToCache would result in a maximum 250,000 stat cache entries and 65,000 file cache entries.

During configuration, you can specify the maxFilesToCache, maxStatCache, and pagepool parameters that control how much cache is dedicated to GPFS. These values can be changed later, so experiment with larger values to find the optimum cache size that improves GPFS performance without affecting other applications.

The mmchconfig command can be used to change the values of maxFilesToCache, maxStatCache, and pagepool. The pagepool parameter is the only one of these parameters that may be changed while the GPFS daemon is running. A pagepool change occurs immediately when using the -i option on the mmchconfig command. Changes to the other values are effective only after the daemon is restarted.

Maximum file system block size allowed
The valid values for the maximum block size for file systems to be created for the nodeset are 16 KB, 64 KB, 256 KB, 512 KB, and 1024 KB (1 MB is also acceptable). After you have configured GPFS, any attempt to create a file system with a block size larger than the maximum block size will fail. See “File system sizing” on page 21 for a discussion of block size values when creating a file system before you make a decision on setting the maximum block size allowed.

DMAPI configuration options
For a discussion of the DMAPI configuration options, see the General Parallel File System for AIX 5L: AIX Clusters Data Management API Guide:

- dmapiEventTimeout
- dmapiSessionFailureTimeout
- dmapiMountTimeout

A sample nodeset configuration
To create a nodeset with these configuration options, allowing all other values to default:

```
/u/gpfsadmin/nodesGPFS1
-A
set1
```

the file containing the list of 32 nodes to be included in the nodeset

Automatically start the GPFS daemon when the nodes come up

the nodeset identifier
-p 100M
pagepool of 100 MB

Issue the command:
`mmconfig -n /u/gpfsadmin/nodesGPFS1 -A -C set1 -p 100M`

To confirm the nodeset configuration, enter:
`mmlsconfig -C set1`

The system displays information similar to:

Configuration data for nodeset set1:
------------------------------------
clusterType rpd
comm_protocol TCP
multinode yes
autoload yes
useSingleNodeQuorum no
pagepool 100M
group Gpfs.set1
recgroup GpfsRec.set1

File systems in nodeset set1:
----------------------------
(none)

---

**File system creation considerations**

File system creation involves anticipating usage within the file system and considering your hardware configurations. Your GPFS file system is created by issuing the `mmcrfs` command. Table 4 details the file system creation options specified on the `mmcrfs` command, which options can be changed later with the `mmchfs` command, and what the default values are.

**Table 4. File system creation options**

<table>
<thead>
<tr>
<th>Option</th>
<th>mmcrfs</th>
<th>mmchfs</th>
<th>default value</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Automatic mount</em> on page 21</td>
<td>X</td>
<td>X</td>
<td>yes</td>
</tr>
<tr>
<td><em>Estimated node count</em> on page 21</td>
<td>X</td>
<td>this value cannot be changed</td>
<td>32</td>
</tr>
<tr>
<td><em>Block size</em> on page 21</td>
<td>X</td>
<td>this value cannot be changed</td>
<td>256K</td>
</tr>
<tr>
<td><em>Maximum number of files</em> on page 22</td>
<td>X</td>
<td>X</td>
<td>file system size/1 MB</td>
</tr>
<tr>
<td>Default metadata replicas, see <em>File system recoverability parameters</em> on page 22</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Maximum metadata replicas, see <em>File system recoverability parameters</em> on page 22</td>
<td>X</td>
<td>this value cannot be changed</td>
<td>1</td>
</tr>
<tr>
<td>Default data replicas, see <em>File system recoverability parameters</em> on page 22</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Maximum data replicas, see <em>File system recoverability parameters</em> on page 22</td>
<td>X</td>
<td>this value cannot be changed</td>
<td>1</td>
</tr>
<tr>
<td><em>Automatic quota activation</em> on page 23</td>
<td>X</td>
<td>X</td>
<td>no</td>
</tr>
<tr>
<td><em>Disk verification</em> on page 24</td>
<td>X</td>
<td></td>
<td>yes</td>
</tr>
</tbody>
</table>
Automatic mount

Whether or not to automatically mount a file system when the GPFS daemon starts may be specified at file system creation by using the -A option on the `mmcrfs` command or changed at a later time by using the -A option on the `mmchfs` command. The default is to have the file system automatically mounted, assuring file system availability whenever the system and disks are up.

Estimated node count

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

When creating a GPFS file system, over estimate the number of nodes that will mount the file system. This input is used in the creation of GPFS data structures that are essential for achieving the maximum degree of parallelism in file system operations (see Appendix A, “GPFS architecture”, on page 45). Although a larger estimate consumes a bit more memory, insufficient allocation of these data structures can limit node ability to process certain parallel requests efficiently, such as the allotment of disk space to a file. If you cannot anticipate the number of nodes, allow the default value to be applied. Specify a larger number if you expect to add nodes, but avoid wildly overestimating as this can affect buffer operations. *This value cannot be changed later.*

File system sizing

Before creating a file system, consider how much data will be stored and how great the demand for the files in the system will be. Each of these factors can help you to determine how much disk resource to devote to the file system, which block size to choose, where to store data and metadata, and how many replicas to maintain.

Block size

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

GPFS offers five block sizes for file systems: 16 KB, 64 KB, 256 KB, 512 KB, and 1024 KB. This value should be specified with the character K, for example 512K. You should choose the block size based on the application set that you plan to support and if you are using RAID hardware.

---

Table 4. File system creation options (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>mmcrfs</th>
<th>mmchfs</th>
<th>default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable DMAPI on page 24</td>
<td>X</td>
<td>X</td>
<td>no</td>
</tr>
<tr>
<td>Mountpoint directory on page 24</td>
<td>X</td>
<td>X</td>
<td>none</td>
</tr>
<tr>
<td>Device name of the file system on page 25</td>
<td>X</td>
<td>this attribute cannot be changed</td>
<td>none</td>
</tr>
<tr>
<td>Disks for the file system on page 25</td>
<td>X</td>
<td>use <code>mmaddisk</code> and  <code>mmdeldisk</code> respectively to add or delete disks from the file system</td>
<td>none</td>
</tr>
<tr>
<td>Nodeset to which the file system belongs on page 25</td>
<td>X</td>
<td>X</td>
<td>the nodeset from which the <code>mmcrfs</code> command is issued</td>
</tr>
</tbody>
</table>

Notes:
1. X – indicates the option is available on the command
2. an empty cell – indicates the option is not available on the command
The 256 KB block size is the default block size and normally is the best block size for file systems that contain large files accessed in large reads and writes.

The 16 KB block size optimizes use of disk storage at the expense of large data transfers.

The 64 KB block size offers a compromise. It makes more efficient use of disk space than 256 KB while allowing faster I/O operations than 16 KB.

The 512 KB and 1024 KB block size may be more efficient if data accesses are larger than 256 KB.

If you plan to use SSA RAID devices in your file system, a larger block size may be more effective and help you to avoid the penalties involved in small block write operations to RAID devices. For example, in a RAID configuration utilizing 4 data disks and 1 parity disk (a 4+P configuration), which utilizes a 64 KB stripe size, the optimal file system block size would be 256 KB (4 data disks \times 64 \text{ KB stripe size} = 256 \text{ KB}). A 256 KB block size would result in a single data write that encompassed the 4 data disks and a parity write to the parity disk. If a block size smaller than 256 KB, such as 64 KB, was used, write performance would be degraded. A 64 KB block size would result in a single disk writing 64 KB and a subsequent read from the three remaining disks in order to compute the parity that is then written to the parity disk. The extra read degrades performance.

The maximum GPFS file system size that can be mounted is limited by the control structures in memory required to maintain the file system. These control structures, and consequently the maximum mounted file system size, are a function of the block size of the file system.

- If your file systems have a 16 KB block size, you may have one or more file systems with a total size of 1 TB mounted.
- If your file systems have a 64 KB block size, you may have one or more file systems with a total size of 10 TB mounted.
- If your file systems have a 256 KB or greater block size, you may have file systems mounted with a total size of not greater than 200 TB where no single file system exceeds 100 TB.

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

The block size is the largest contiguous amount of disk space allocated to a file and therefore the largest amount of data that can be accessed in a single I/O operation. The subblock is the smallest unit of disk space that can be allocated. For a block size of 256 KB, GPFS reads as much as 256 KB of data in a single I/O operation and small files can occupy as little as 8 KB of disk space. With a block size of 16 KB, small files occupy as little as 512 bytes of disk space (not counting the inode), but GPFS is unable to read more than 16 KB in a single I/O operation.

**Maximum number of files**
The maximum number of files in a file system may be specified at file system creation by using the `-N` option on the `mmcrfs` command or changed at a later time by using the `-F` option on the `mmchfs` command. This value defaults to the size of the file system at creation divided by 1 MB and cannot exceed the architectural limit of 256 million.

These options limit the maximum number of files that may actively exist within the file system. However, the maximum number of files in the file system is never allowed to consume all of the file system space and is thus restricted by the formula:

\[
\text{maximum number of files} = \frac{(\text{total file system space} / 2)}{(\text{inode size} + \text{subblock size})}
\]

**File system recoverability parameters**
The metadata (inodes, directories, and indirect blocks) and data replication parameters are set at the file system level and apply to all files. They are initially set for the file system when issuing the `mmcrfs` command.
command. They can be changed for an existing file system using the `mmchfs` command, but modifications only apply to files subsequently created. To apply the new replication values to existing files in a file system, issue the `mmrestripes` command.

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

**Default metadata Replicas**
The default number of copies of metadata for all files in the file system may be specified at file system creation by using the `-m` option on the `mmcrfs` command or changed at a later time by using the `-m` option on the `mmchfs` command. This value must be equal to or less than `MaxMetadataReplicas`, and cannot exceed the number of failure groups with disks that can store metadata. The allowable values are 1 or 2, with a default of 1.

**Maximum metadata replicas**
The maximum number of copies of metadata for all files in the file system may be specified at file system creation by using the `-M` option on the `mmcrfs` command or allowed to default to 1. The allowable values are 1 or 2, but it cannot be lower than `DefaultMetadataReplicas`. This value can only be overridden by a system call when the file has a length of 0.

**Default data replicas**
The default replication factor for data blocks may be specified at file system creation by using the `-r` option on the `mmcrfs` command or changed at a later time by using the `-r` option on the `mmchfs` command. This value must be equal to or less than `MaxDataReplicas`, and the value cannot exceed the number of failure groups with disks that can store data. The allowable values are 1 and 2, with a default of 1.

**Maximum data replicas**
The maximum number of copies of data blocks for a file may be specified at file system creation by using the `-R` option on the `mmcrfs` command or allowed to default to 1. The allowable values are 1 and 2, but cannot be lower than `DefaultDataReplicas`. This value can only be overridden by a system call when the file has a length of 0.

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

The GPFS quota system helps you to control the allocation of files and data blocks in a file system. GPFS quotas can be defined for individual users or groups of users. Quotas should be installed by the system administrator if control over the amount of space used by the individual users or groups of users is desired. When setting quota limits for a file system, the system administrator should consider the replication factors of the file system. GPFS quota management takes replication into account when reporting on and determining if quota limits have been exceeded for both block and file usage. In a file system which has either type of replication set to a value of two, the values reported on by both the `mmlsquota` and the `mmrepquota` commands are double the value reported by the `ls` command.

GPFS quotas operate with three parameters that you can explicitly set using the `mmedquota` and `mmdefedquota` commands:

1. Soft limit
2. Hard limit
3. Grace period
The soft limits define levels of disk space and files below which the user or group can safely operate. The hard limits define the maximum disk space and files the user or group can accumulate. Specify hard and soft limits for disk space in units of kilobytes (k or K) or megabytes (m or M). If no suffix is provided, the number is assumed to be in bytes.

The grace period allows the user or group to exceed the soft limit for a specified period of time (the default period is one week). If usage is not reduced to a level below the soft limit during that time, the quota system interprets the soft limit as the hard limit and no further allocation is allowed. The user or group can reset this condition by reducing usage enough to fall below the soft limit.

**Default quotas**

Applying default quotas ensures all new users or groups of users of the file system will have minimum quota limits established. If default quota values for a file system are not enabled, a new user or group has a quota value of zero which establishes no limit to the amount of space they can use. Default quotas may be set for a file system only if the file system was created with the `-Q yes` option on the `mmcrfs` or updated with the `-Q` option on the `mmchfs` command. Default quotas may then be enabled for the file system by issuing the `mmdefquotaon` command and default values established by issuing the `mmdefedquota` command.

**Quota system files**

The quota system maintains usage and limits data in the `user.quota` and `group.quota` files that reside in the root directories of GPFS file systems. These files are built with the information provided in the `mmedquota` and `mmdefedquota` commands. These files are updated through normal allocation operations throughout the file system and when the `mmcheckquota` command is issued. The `user.quota` and `group.quota` files are readable by the `mmlsquota` and `mmrepquota` commands.

These files are also read when mounting a file system with quotas enabled. If these files are not available when mounting the file system, new quota files are created. If the files exist in the file system’s root directory, there are 3 possible situations:

1. The files contain quota information and the user wants these files to be used.
2. The files contain quota information, however, the user wants different files to be used. In order to specify the usage of different files, the `mmcheckquota` command must be issued prior to the `mount` of the file system.
3. The files do not contain quota information, but are used during the mount of the file system. In this case the `mount` will fail and appropriate error messages will be displayed. See the General Parallel File System for AIX 5L: AIX Clusters Problem Determination Guide for further information regarding `mount` failures.

**Disk verification**

When you create your file system, you may check to ensure the disks you are specifying do not already belong to an existing file system by using the `-v` option on the `mmcrfs` command. The default is to verify disk usage. You should only specify `no` when you want to reuse disks that are no longer needed for an existing GPFS file system. To determine which disks are no longer in use by any file system, issue the `mmlsgpfsdisk -F` command.

**Enable DMAPI**

Whether or not the file system can be monitored and managed by the GPFS Data Management API (DMAPI) may be specified at file system creation by using the `-z` option on the `mmcrfs` command or changed at a later time by using the `-z` option on the `mmchfs` command. The default is not to enable DMAPI for the file system. For further information on DMAPI for GPFS, see General Parallel File System for AIX 5L: AIX Clusters Data Management API Guide.

**Mountpoint directory**

There is no default mountpoint directory supplied for the file system. You must specify the directory.
Device name of the file system

Specify a device name for your file system that is unique across all GPFS nodesets and is not an existing entry in /dev. The device name need not be fully qualified. fs0 is as acceptable as /dev/fs0. The file system name cannot be changed at a later time.

Disks for the file system

Prior to issuing the mmcrfs command you must decide if you will:

1. Create new disks via the mmcrlv command.
2. Select disks previously created by the mmcrlv command, but no longer in use in any file system.
3. Use the rewritten disk descriptor file produced by the mmcrlv command or create a new list of disk descriptors. When using the rewritten file, the Disk Usage and Failure Group specifications will remain the same as specified on the mmcrlv command.

When issuing the mmcrfs command you may either pass the disk descriptors in a file or provide a list of disk descriptors to be included. The file eliminates the need for command line entry of these descriptors using the list of DiskDescs. You may use the rewritten file created by the mmcrlv command, or create your own file. When using the file rewritten by the mmcrlv command, the Disk Usage and Failure Group values are preserved. Otherwise, you must specify a new value or accept the default. You can use any editor to create such a file to save your specifications. When providing a list on the command line, each descriptor is separated by a semicolon (;) and the entire list must be enclosed in quotation marks (' or "). The current maximum number of disk descriptors that can be defined for any single file system is 1024.

Each disk descriptor must be specified in the form (second and third fields reserved):

DiskName:::DiskUsage:FailureGroup

DiskName

You must specify the logical volume name. For details on creating a logical volume, see "Logical volume creation considerations" on page 11. To use an existing logical volume in the file system, only the logical volume name need be specified in the disk descriptor. The disk name must be set up the same on all nodes in the nodeset.

Disk Usage

Specifies what is to be stored on the disk. Specify one or accept the default:

- dataAndMetadata (default)
- dataOnly
- metadataOnly

Failure Group

A number identifying the failure group to which this disk belongs. You can specify any value from -1 (where -1 indicates that the disk has no point of failure in common with any other disk) to 4000. If you do not specify a failure group, the value defaults to the -1. GPFS uses this information during data and metadata placement to assure that no two replicas of the same block are written in such a way as to become unavailable due to a single failure. All disks that are attached to the same disk adapter, should be placed in the same failure group.

Nodeset to which the file system belongs

If you do not specify a particular nodeset for the file system to belong to, by default it will belong to the nodeset from which the mmcrfs command was issued. You may move a file system to a different nodeset by issuing the mmchfs command. When moving a file system, before the file system is mounted on the target nodeset, it must have the same disk connectivity as the original nodeset.
A sample file system creation

To create a file system with these configuration options, allowing all other values to default:

A  Automatically mount the file system when the GPFS daemon starts.
R  Default maximum number of copies of data blocks for the file.
M  Default maximum number of copies of inodes, directories, and indirect blocks for the file.
v  Verify the specified disks do not belong to an existing file system.

Issue the command:

\texttt{mmcrfs /fs3 fs3 -F crlvdd3 -A yes -R 2 -M 2 -v yes}

The system displays information similar to:

\texttt{mmcrfs: 6027-1371 Propagating the changes to all affected nodes. This is an asynchronous process.}

To confirm the file system configuration, issue the command:

\texttt{mmlsfs fs3}

The system displays information similar to:

\begin{center}
\begin{tabular}{ll}
\hline
flag & value \\
\hline
-s & roundRobin \\
-f & 8192 \\
-i & 512 \\
-I & 16384 \\
-m & 1 \\
-M & 2 \\
r & 1 \\
-R & 2 \\
a & 1048576 \\
n & 32 \\
-B & 262144 \\
-Q & none \\
-F & 33792 \\
-V & 6.00 \\
-z & no \\
-d & gpfs33lv \\
-A & yes \\
-C & set1 \\
-E & no \\
-S & no \\
-o & none \\
\hline
\end{tabular}
\end{center}

flag value description

- roundRobin  Stripe method
- 8192 Minimum fragment size in bytes
- 512 Inode size in bytes
- 16384 Indirect block size in bytes
- 1 Default number of metadata replicas
- 2 Maximum number of metadata replicas
- 1 Default number of data replicas
- 2 Maximum number of data replicas
- 1048576 Estimated average file size
- 32 Estimated number of nodes that will mount file system
- 262144 Block size
- none Quotas enforced
- Default quotas enabled
- 33792 Maximum number of inodes
- 6.00 File system version. Highest supported version: 6.00
- Is DMAPI enabled?
- Disks in file system
- Automatic mount option
- GPFS nodeset identifier
- Exact mtime default mount option
- Suppress atime default mount option
- Additional mount options
Part 2. Preparing your system for GPFS

Part 2 provides information on:

- Chapter 3, “Installing GPFS”, on page 29
- Chapter 4, “Tuning your system for GPFS”, on page 35
- Chapter 5, “Migration, coexistence, and compatibility”, on page 37
- Chapter 6, “Permanently uninstalling GPFS”, on page 41
Chapter 3. Installing GPFS

It is suggested you read Chapter 2, “Planning for GPFS”, on page 9 before getting started.

Do not attempt to install GPFS if you do not have the correct hardware (“Hardware specifications” on page 9) and software (“Programming specifications” on page 9) prerequisites installed on your system (see “Verifying the level of prerequisite software” on page 30).

Ensure that the PATH environment variable on each node includes:
• /usr/lpp/mmfs/bin

The installation process includes:
1. “Electronic license agreement”
2. “Files to ease the installation process”
3. “Verify there is no conflicting software installed”
4. “Verifying the level of prerequisite software” on page 30
5. “Installation procedures” on page 31
6. “Verifying the GPFS installation” on page 33
7. “What’s next after completing the installation of GPFS” on page 33

Electronic license agreement

Beginning with GPFS 2.1, software license agreements are shipped and viewable electronically. If a product has an electronic license agreement, it must be accepted before software installation can continue.

For additional software package installations, the installation cannot occur unless the appropriate license agreements are accepted. When using the installp command, use the -Y flag to accept licenses and the -E flag to view license agreement files on the media.

Files to ease the installation process

Creation of a file that contains all of the nodes in your GPFS cluster prior to the installation of GPFS will be useful during the installation process. Using either hostnames or IP addresses when constructing the file will allow you to use this information when creating your cluster via the mmcrcluster command.

Create a file listing the nodes which will be running GPFS, one per line:

```
/tmp/gpfs.allnodes
k145n01.dpd.ibm.com
k145n02.dpd.ibm.com
k145n03.dpd.ibm.com
k145n04.dpd.ibm.com
k145n05.dpd.ibm.com
k145n06.dpd.ibm.com
k145n07.dpd.ibm.com
k145n08.dpd.ibm.com
```

Verify there is no conflicting software installed

When installing GPFS for the first time, you must verify there is no conflicting software already installed. Due to common components shared by GPFS, IBM Multi-Media Server, and IBM Video Charger, the kernel extensions for GPFS cannot coexist with these products on the same system. If either of these products exist on your system, you will have to remove them in order to successfully install GPFS. Run this short piece of code:
#!/usr/bin/ksh
for node in $(cat /tmp/gpfs.allnodes)
  do
      rsh $node lslpp -l "mmfs.*"
  done

If any `mmfs` filesets exist, you have either one or both of the IBM Multi-Media Server and IBM Video Charger products installed and you should remove them.

Verifying the level of prerequisite software

It is necessary to verify you have the correct levels of the prerequisite software installed. If the correct level of prerequisite software is *not* installed, see the appropriate installation manual before proceeding with your GPFS installation:

**Note:** When installing AIX from system images created with the `mksysb` command, duplicate node ids may be generated on those nodes. The `lsnodedid` (available in `/usr/sbin/rsct/bin`) has been provided for you to verify whether or not node ids are duplicated within the cluster. If a duplicate node id is found, go to the RSCT Resource Monitoring and Control README located at [www.ibm.com/servers/eserver/clusters/library](http://www.ibm.com/servers/eserver/clusters/library) and follow the procedure to generate unique node ids.

1. AIX 5L Version 5.1 with APAR IY33002, or later modifications:
   ```
   lslpp -l bos.mp
   ```

2. For a GPFS cluster type `rpd`, the following RSCT file sets must be installed on each node in the GPFS cluster:
   ```
   lslpp -l rsct*
   ```

   The system displays information similar to:
   
<table>
<thead>
<tr>
<th>Fileset</th>
<th>Level</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path: /usr/lib/objrepos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rsct.basic.rte</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Basic Function</td>
</tr>
<tr>
<td>rsct.compat.basic.rte</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Event Management Basic Function</td>
</tr>
<tr>
<td>rsct.compat.clients.rte</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Event Management Client Function</td>
</tr>
<tr>
<td>rsct.core.auditrm</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Audit Log Resource Manager</td>
</tr>
<tr>
<td>rsct.core.ermr</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Event Response Resource Manager</td>
</tr>
<tr>
<td>rsct.core.fsrm</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT File System Resource Manager</td>
</tr>
<tr>
<td>rsct.core.hostrm</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Host Resource Manager</td>
</tr>
<tr>
<td>rsct.core.mmc</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Resource Monitoring and Control</td>
</tr>
<tr>
<td>rsct.core.sec</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Security</td>
</tr>
<tr>
<td>rsct.core.sr</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Registry</td>
</tr>
<tr>
<td>rsct.core.utils</td>
<td>2.2.1.20</td>
<td>COMMITTED</td>
<td>RSCT Utilities</td>
</tr>
</tbody>
</table>

   Path: /etc/objrepos

   | rsct.basic.rte    | 2.2.1.20| COMMITTED  | RSCT Basic Function                |
   | rsct.compat.basic.rte | 2.2.1.20| COMMITTED  | RSCT Event Management Basic Function |
   | rsct.core.mmc     | 2.2.1.20| COMMITTED  | RSCT Resource Monitoring and Control |

3. For a GPFS cluster type `hacmp`, HACMP/ES Version 4 Release 4.1, or later modifications must be installed on each node in the GPFS cluster:
   ```
   lslpp -l cluster*
   ```

   The system displays information similar to:

<table>
<thead>
<tr>
<th>Fileset</th>
<th>Level</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path: /usr/lib/objrepos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cluster.doc.en_US.es.html</td>
<td>4.4.1.2</td>
<td>COMMITTED</td>
<td>HAES Web-based HTML Documentation - U.S. English</td>
</tr>
<tr>
<td>cluster.doc.en_US.es.pdf</td>
<td>4.4.1.1</td>
<td>COMMITTED</td>
<td>HAES PDF Documentation - U.S. English</td>
</tr>
<tr>
<td>cluster.doc.en_US.es.ps</td>
<td>4.4.1.0</td>
<td>COMMITTED</td>
<td>HAES Postscript Documentation - U.S. English</td>
</tr>
</tbody>
</table>
Installation procedures

Follow these steps to install the GPFS software using the installp command. This procedure installs GPFS on one node at a time.

**Note:** The installation procedures are generalized for all levels of GPFS. Ensure you substitute the correct numeric value for the modification (m) and fix (f) levels, where applicable. The modification and fix level are dependent upon the level of PTF support.

Creating the GPFS directory

On any node (normally node 1), create a subdirectory in /tmp/gpfs1pp with the command:

```
mkdir /tmp/gpfs1pp
```

Then copy the installation images from the CD-ROM to the new directory, using the `bffcreate` command:

```
bffcreate -qvX -t /tmp/gpfs1pp -d /dev/cd0 all
```

This will place the following GPFS images in the image directory:

1. mmfs.base.usr.3.5.m.f
2. mmfs.gpfs.usr.2.1.m.f
3. mmfs.msg.en_US.usr.3.5.m.f
4. mmfs.gpfsdocs.data.3.5.m.f

Installing the GPFS man pages

There are two sets of man pages shipped with the GPFS for AIX 5L program product. There is one set for the PSSP cluster environment and one set for the AIX cluster environment. You should set your `MANPATH` environment variable to access the correct set of man pages (see "MANPATH environment variable" on page 36).

**Note:** `mmfs.gpfsdocs.data` need not be installed on all nodes if man pages are not desired or local file system space on the node is minimal.
Creating the GPFS installation images

1. Make the new image directory the current directory:
   `cd /tmp/gpfslpp`

2. Use the `inutoc` command to create a `.toc` file. The `.toc` file is used by the `installp` command.
   `inutoc .`

3. To view the product README after creating the installation images:
   `installp -l -d . mmfs | more`

If you have previously installed GPFS on your system, during the install process you may see messages similar to:

Some configuration files could not be automatically merged into the system during the installation. The previous versions of these files have been saved in a configuration directory as listed below. Compare the saved files and the newly installed files to determine if you need to recover configuration data. Consult product documentation to determine how to merge the data.

Configuration files which were saved in /lpp/save.config:
   /var/mmfs/etc/gpfsready
   /var/mmfs/etc/mmfs.cfg
   /var/mmfs/etc/mmfsdown.scr
   /var/mmfs/etc/mmfsup.scr

If you have made changes to any of these files, you will have to reconcile the differences with the new versions of the files in directory /var/mmfs/etc. This does not apply to file /var/mmfs/etc/mmfs.cfg which is automatically maintained by GPFS.

Installing GPFS on your network

Install GPFS according to one of the following procedures, depending on whether or not your network has a shared file system.

Installing on a shared file system network

1. Ensure that the image directory is NFS-exported to the system nodes.

   Export the local directory, which is not in the `/etc/exports` file, without restrictions for NFS clients to mount:
   `exportfs -i /tmp/gpfslpp`

   To display whether or not k145n01 has any exported directories, enter:
   `showmount -e k145n01`

   The system displays information similar to:
   `export list for k145n01:
   /tmp/gpfslpp (everyone)`

2. On each node, issue a `mount` command to NFS mount the image directory:
   `mount k145n01:/tmp/gpfslpp /mnt`

3. On the first node in the GPFS nodeset, issue an `installp` command to install GPFS:
   `installp -agXYd /tmp/gpfslpp all`

4. To install GPFS on the rest of the nodes individually, issue an `installp` command on each of the nodes:
   `installp -agXYd /mnt all`
Installing on a non-shared file system network
If the GPFS installation directory is not in a shared network file system: copy the images to each node, from the node that the GPFS installation directory was initially installed on (for example k145n01), as follows:

```
mkdir /tmp/gpfslpp
rcp -p k145n01:/tmp/gpfslpp/* /tmp/gpfslpp
```

Then, install on each node from its local GPFS installation directory:

```
installp -agXy /tmp/gpfslpp all
```

Verifying the GPFS installation
Use the **lslpp** command to verify the installation of GPFS filesets on each system node:

```
kl45n01:/ # lslpp -l mmfs*
```

Output similar to the following should be returned:

<table>
<thead>
<tr>
<th>Fileset</th>
<th>Level</th>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path: /usr/lib/objrepos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mmfs.base.cmds</td>
<td>3.5.0.0</td>
<td>COMMITTED</td>
<td>GPFS File Manager Commands</td>
</tr>
<tr>
<td>mmfs.base.rte</td>
<td>3.5.0.0</td>
<td>COMMITTED</td>
<td>GPFS File Manager</td>
</tr>
<tr>
<td>mmfs.gpfs.rte</td>
<td>2.1.0.0</td>
<td>COMMITTED</td>
<td>GPFS File Manager</td>
</tr>
<tr>
<td>mmfs.msg.en_US</td>
<td>3.5.0.0</td>
<td>COMMITTED</td>
<td>GPFS Server Messages - U.S. English</td>
</tr>
<tr>
<td>Path: /etc/objrepos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mmfs.base.rte</td>
<td>3.5.0.0</td>
<td>COMMITTED</td>
<td>GPFS File Manager</td>
</tr>
<tr>
<td>mmfs.gpfs.rte</td>
<td>2.1.0.0</td>
<td>COMMITTED</td>
<td>GPFS File Manager</td>
</tr>
<tr>
<td>Path: /usr/share/lib/objrepos</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mmfs.gpfsdocs.data</td>
<td>3.5.0.0</td>
<td>COMMITTED</td>
<td>GPFS Server Manpages</td>
</tr>
</tbody>
</table>

What’s next after completing the installation of GPFS
Now that you have successfully installed GPFS, you must (in the specified order):

1. Tune your system prior to configuring GPFS (see Chapter 4, "Tuning your system for GPFS", on page 35).
2. Create a GPFS cluster, see "GPFS cluster creation considerations" on page 14.
3. Configure your GPFS nodeset by issuing the **mmconfig** command (see "Nodeset configuration considerations" on page 15).
4. Start GPFS by issuing the **mmstartup** command (see the *General Parallel File System for AIX 5L: AIX Clusters Administration and Programming Reference*).
5. Create your file systems by issuing the **mmcrfs** command (see "File system creation considerations" on page 20).
Chapter 4. Tuning your system for GPFS

Before you configure GPFS (see “Nodeset configuration considerations” on page 15), you need to configure and tune your system. Values suggested here reflect evaluations made on the hardware available at the time this document was written. For the latest values regarding new hardware, see www.ibm.com/servers/eserver/pseries/software/sp/gpfs.html.

System configuration settings

The settings on your system should be applied prior to configuring GPFS. The particular components whose configuration settings need to be considered are:

1. “Security”
2. “Topology Services”
3. “Communications I/O”
4. “Disk I/O” on page 36
5. “nofiles” on page 36
6. “MANPATH environment variable” on page 36

Security

When using rcp and rsh for remote communication, a properly configured .rhosts file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the mmcrcluster or the mmchcluster, you must ensure:

- Proper authorization is granted to all nodes in the GPFS cluster.
- The nodes in the GPFS cluster can communicate without the use of a password.

If this has not been properly configured, you will get GPFS errors.

Topology Services

GPFS requires invariant network connections. An adapter with an invariant address is one that cannot be used for IP address takeover operations. The adapter must be part of a network with no service addresses and should not have a standby adapter on the same network. That is, the port on a particular IP address must be a fixed piece of hardware that is translated to a fixed network adapter and is monitored for failure. Topology Services should be configured to heartbeat over this invariant address. For information on configuring Topology Services:

1. For a cluster type of rpd, see the Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference
2. For a cluster type of hacmp, see the High Availability Cluster Multi-Processing for AIX: Enhanced Scalability Installation and Administration Guide

Communications I/O

The ipqmaxlen network option should be considered when configuring for GPFS. The ipqmaxlen parameter controls the number of incoming packets that can exist on the IP interrupt queue. The default of 128 is often insufficient. The recommended setting is 512.

no -o ipqmaxlen=512

Since this option must be modified at every reboot, it is suggested it be placed at the end of one of the system start-up files, such as the /etc/rc.net shell script. For detailed information on the ipqmaxlen parameter, see the AIX 5L Performance Management Guide.
Disk I/O
The disk I/O option to consider when configuring GPFS and using SSA RAID:

max_coalesce
The max_coalesce parameter of the SSA RAID device driver allows the device driver to coalesce requests which have been broken up to satisfy LVM requirements. This parameter can be critical when using RAID and is required for effective performance of RAID writes. The recommended setting is 0x40000 for 4+P RAID.

- To view:
  `lsattr -E -l hdiskX -a max_coalesce`
- To set:
  `chdev -l hdiskX -a max_coalesce=0x40000`

For further information on the max_coalesce parameter see the [AIX 5L Technical Reference: Kernel and Subsystems, Volume 2](#).

nofiles
Ensure that nofiles, the file descriptor limit in `/etc/security/limits`, is set to -1 (unlimited) on the Control Workstation.

MANPATH environment variable
Multiple sets of GPFS man pages are shipped with the GPFS for AIX 5L program product. There is one set for the PSSP cluster environment and one set for the AIX cluster environment. In order to access the correct set of man pages for the AIX cluster environment, you must set your MANPATH environment variable to include `/usr/lpp/mmfs/gpfsdocs/man/aix`. 
Chapter 5. Migration, coexistence, and compatibility

This chapter contains information to help you migrate your GPFS nodeset:

- **Migrating to GPFS 2.1**
- **Coexistence** on page 39
- **Compatibility** on page 40

Migrating to GPFS 2.1

These conventions must be followed when migrating to GPFS 2.1:

- All nodes within a GPFS nodeset must be upgraded to GPFS 2.1 at the same time.
- New file system functions existing in GPFS 2.1 are not usable until you explicitly authorize these changes by issuing the `mmchfs -V` command.
- After completing the migration, before you can use an existing logical volume which was not part of any GPFS file system at the time of migration, you must:
  1. Export the logical volume
  2. Recreate the logical volume using the `mmcrlv` command.
  3. Add the logical volume to a file system
- In order to use the 64-bit versions of the GPFS programming interfaces, you must recompile your code using the appropriate 64-bit options for your compiler.
- The GPFS Data Management API is only supported in a 32-bit version.

The migration process includes:

- **GPFS nodesets for migration**
- **Staged migration to GPFS 2.1**
- **Full migration to GPFS 2.1** on page 38
- **Reverting to the previous level of GPFS** on page 39

GPFS nodesets for migration

You have the ability to define more than one GPFS nodeset in the same GPFS cluster (see **Nodeset configuration considerations** on page 15). This allows you to create a separate nodeset for testing the new level of GPFS code without affecting the main GPFS nodeset running at production level.

Your system should have a minimum of 6 nodes in order to use multiple GPFS nodesets and do a staged migration. If you have less than 6 nodes, it is recommended you do a full migration. See **Full migration to GPFS 2.1** on page 38

Staged migration to GPFS 2.1

In a staged migration you will first install the new level of GPFS only on a small subset of nodes as a test nodeset. Once you are satisfied with the new code, you can upgrade the rest of the nodes.

A staged migration to the new level of GPFS consists of the following steps:

1. Decide which nodes will be used to test the new level of GPFS. It is recommended that you use at least three nodes for this purpose. Create a file, `/tmp/gpfs.nodes` containing the list of hostnames, one per line, of the nodes to be used in the migration.

   The hostname or IP address must refer to the communications adapter. Alias interfaces are not allowed. Use the original address or a name that is resolved by the `host` command to that original address. You may specify a node using any of these forms:
2. Stop GPFS on all the nodes in the test nodeset:
   
   mmshutdown -W /tmp/gpfs.nodes

3. Using the `mmdelnode` command, delete the nodes in the test nodeset from the main GPFS nodeset. See the General Parallel File System for AIX 5L: AIX Clusters Administration and Programming Reference and search on deleting nodes from a GPFS nodeset.

4. Copy the install images as described in “Creating the GPFS directory” on page 31. Install the new code on the nodes in the test nodeset. The install process will not affect your main GPFS nodeset. See Chapter 3, “Installing GPFS”, on page 29.

5. Reboot all the nodes in the test nodeset. This is required so the kernel extensions can be replaced.

6. Using the `mmconfig` command, create the test nodeset. See “Nodeset configuration considerations” on page 15.

7. Using the `mmcrfs` command, create a file system for testing the new level of GPFS (see “File system creation considerations” on page 20).

   Notes:
   a. If you want to use an existing file system, move the file system by issuing the `mmchfs` command with the `-C` option.
   b. If the file system was created under your original level of GPFS, you must explicitly migrate the file system (`mmchfs -V`) before you can use the new functions in the latest level of GPFS. Remember you cannot go back once you do this step! Any attempt to mount a migrated file system on a back-level GPFS system will be rejected with an error. See the General Parallel File System for AIX 5L: AIX Clusters Administration and Programming Reference for complete information on the GPFS administration commands.

8. Operate with the new level of code for awhile to make sure you want to migrate the rest of the nodes. If you decide to go back to your original GPFS level, see “Reverting to the previous level of GPFS” on page 39.

9. Attention: You cannot go back once you do this step! Any attempt to mount a migrated file system on a back-level GPFS system will be rejected with an error.

Once you have decided to permanently accept the latest level of GPFS, for each of the file systems that are in the new nodeset, issue:

```
mmchfs filesystem -V
```

You may now exploit the new functions in the GPFS code.

10. When you are ready to migrate the rest of the nodes in the main GPFS nodeset:
   a. Follow steps 2, 4, 5, and 9.
   b. Either delete the file systems from the test nodeset by issuing the `mmdelfs` command, or move them to the main GPFS nodeset by issuing the `mmchfs` command with the `-C` option.
   c. Delete the nodes from the test nodeset by issuing the `mmdelnodes` command and add them to the main nodeset by issuing the `mmaddnode` command.

11. Issue the `mmlsfs` command to verify that the file system has been upgraded to latest level of GPFS. For GPFS 2.1 the `-V` option should indicate a version level of 6.

12. You may now operate with the new level of GPFS code.

**Full migration to GPFS 2.1**
A full migration to the new GPFS level consists of these steps:

1. Copy the install images to all nodes as described in “Creating the GPFS directory” on page 31
2. Stop GPFS on all nodes:
   `mmshutdown -a`

3. Install the new code on all nodes. See Chapter 3, “Installing GPFS”, on page 29

4. Reboot all nodes. This is required so the kernel extensions can be replaced.

5. Operate with the new level of code for awhile to make sure you want to permanently migrate.
   If you decide to go back to the previous level of GPFS, see “Reverting to the previous level of GPFS”.

6. **Attention:** Remember you cannot go back once you do this step! Any attempt to mount a migrated file system on a back-level GPFS system will be rejected with an error.
   Once you have decided to permanently accept the latest level of GPFS, for each of the file systems, issue:
   `mmchfs filesystem -V`

7. You may now operate with the new level of GPFS code.

**Reverting to the previous level of GPFS**

If you should decide not to continue the migration to the latest level of GPFS, and you have not issued the `mmchfs -V` command, you may reinstall the back level of GPFS using the following steps:

1. Copy the install images of the back level GPFS code on all affected nodes.

2. Stop GPFS on all affected nodes:
   `mmshutdown -W /tmp/gpfs.nodes`

3. If you used a test nodeset for testing the latest level of GPFS, return all the nodes in the test nodeset to the main nodeset:
   a. Delete all file systems in the test nodeset that have the latest version number. Use the `mmlsfs -V` command to display the version number of the file system.
   b. Either delete or move to the main GPFS nodeset all file systems that are still at the back level of GPFS.
   c. Use the `mmdelnodeset` command to delete all nodes from the test nodeset.
   d. Use the `mmaddnode` command to add all of the nodes back into the main GPFS nodeset.

4. Run the deinstall program on each node to remove the GPFS 2.1 level of code.
   This program will not remove any customized files:
   `installp -u mmfs`

5. Install the original install images and all required PTFs.

6. Reboot all nodes.

**Coexistence**

GPFS file systems and nodesets must follow these coexistence guidelines:

- A GPFS file system may only be accessed from a single nodeset.
- All nodes in a GPFS nodeset must have been defined to the GPFS cluster.
- 32-bit and 64-bit applications may coexist within a GPFS nodeset.
- It is not possible for different levels of GPFS to coexist in the same nodeset. However, it is possible to run multiple nodesets at different levels of GPFS.

Due to common components shared by GPFS, IBM Multi-Media Server, and IBM Video Charger, the kernel extensions for GPFS cannot coexist with these products on the same system (see “Verify there is no conflicting software installed” on page 29).
The coexistence of an RSCT Peer Domain and PSSP or HACMP on the same node is not supported. See the RSCT Resource Monitoring and Control README located at www.ibm.com/servers/eserver/clusters/library.

---

**Compatibility**

When operating in a 64-bit environment:

- In order to use 64-bit versions of the GPFS programming interfaces created in an AIX 4.3 environment, you must recompile your code for use in an AIX 5L environment. All other applications which executed on the previous release of GPFS will execute on the new level of GPFS.
- GPFS supports interoperability between 32-bit and 64-bit GPFS kernel extensions within a nodeset.

File systems created under the previous release of GPFS may continue to be used under the new level of GPFS. However, once a GPFS file system has been explicitly changed by issuing the `mmchfs` command with the `-V` option, the disk images can no longer be read by a back level file system. You will be required to recreate the file system from the backup medium and restore the content if you choose to go back after this command has been issued.

File systems created for a PSSP or loose cluster (Linux) environment, may not be used in an AIX cluster environment.
Chapter 6. Permanently uninstalling GPFS

GPFS maintains a number of files that contain configuration and file system related data. Since these files are critical for the proper functioning of GPFS and must be preserved across releases, they are not automatically removed when you uninstall GPFS.

Follow these steps if you do not intend to use GPFS any more on any of the nodes in your cluster and you want to remove all traces of GPFS:

**Attention:** After following these steps and manually removing the configuration and file system related information, you will permanently loose access to all of your current GPFS data.

1. **unmount** all GPFS file systems on all nodes.
2. Issue the `mmdelfs` command to delete all GPFS file systems.
3. Issue the `mmshutdown -a` command to shutdown GPFS on all nodes.
4. Issue the `installp -u` command to uninstall all GPFS filesets from all nodes.
5. Remove the `/var/mmfs` and `/usr/lpp/mmfs` directories.
6. Remove all files that start with `mm` from the `/var/adm/ras` directory.
Part 3. Appendixes
Appendix A. GPFS architecture

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

A discussion of GPFS architecture includes:

- Special management functions
- Use of disk storage and file structure within a GPFS file system
- GPFS and memory
- Component interfaces
- Application and user interaction with GPFS
- GPFS command processing
- Recovery
- GPFS cluster data

Special management functions

In general, GPFS performs the same functions on all nodes. It handles application requests on the node where the application exists. This provides maximum affinity of the data to the application. There are three cases where one node provides a more global function affecting the operation of multiple nodes. These are nodes acting as:

1. The GPFS configuration manager
2. The file system manager
3. The metanode

The GPFS configuration manager

There is one GPFS configuration manager per nodeset. The oldest continuously operating node in the nodeset, as monitored by Group Services, is automatically assigned as the GPFS configuration manager. If it should fail for any reason, the next oldest node takes its place.

The configuration manager selects the file system manager node and determines whether or not a quorum of nodes exist. A quorum of nodes is the minimum number of nodes in the GPFS nodeset which must be running in order for the GPFS daemon to start and for file system usage to continue. Quorum is enforced within a nodeset to prevent multiple nodes from assuming the role of file system manager (see Quorum on page 18). Multiple nodes assuming this role would pose potential data corruption as the token management function resides on the file system manager node.

The file system manager

There is one file system manager per file system which handles all of the nodes using the file system. The services provided by the file system manager include:

1. File system configuration
   - Processes changes to the state or description of the file system:
     - Adding disks
     - Changing disk availability
     - Repairing the file system

   Mount and unmount processing is performed on both the file system manager and the node requesting the service.

2. Management of disk space allocation
Controls which regions of disks are allocated to each node, allowing effective parallel allocation of space.

3. Token management

The token management function resides within the GPFS daemon on each node in the nodeset. For each mount point, there is a token management server, which is located at the file system manager. The token management server coordinates access to files on shared disks by granting tokens that convey the right to read or write the data or metadata of a file. This service ensures the consistency of the file system data and metadata when different nodes access the same file. The status of each token is held in two places:

a. On the token management server
b. On the token management client holding the token

The first time a node accesses a file it must send a request to the file system manager to obtain a corresponding read or write token. After having been granted the token, a node may continue to read or write to the same file without requiring additional interaction with the file system manager, until an application on another node attempts to read or write to the same region in the file.

The normal flow for a token is:

• A message to the token management server.

  The token management server then either returns a granted token or a list of the nodes which are holding conflicting tokens.

• The token management function at the requesting node then has the responsibility to communicate with all nodes holding a conflicting token and get them to relinquish the token.

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

4. Quota management

In a quota-enabled file system, the file system manager automatically assumes quota management responsibilities whenever the GPFS file system is mounted. Quota management involves 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. In order to reduce the need for frequent space requests from nodes writing to the file system, more disk blocks are allocated than requested (see "Automatic quota activation" on page 23).

5. Security services

GPFS will use the security enabled for the environment in which it is running, see "Security" on page 35.

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

Depending on the application workload, the memory and CPU requirements for the services provided by the file system manager may make it undesirable to run a resource intensive application on the same node as the file system manager. GPFS allows you to control the pool of nodes from which the file system manager is chosen. When configuring your nodeset or adding nodes to your nodeset, you can specify which nodes are to be made available to this pool of nodes. A node’s designation may be changed at anytime by issuing the `mmchconfig` command. These preferences are honored except in certain failure situations where multiple failures occur (see the General Parallel File System for AIX 5L: AIX Clusters Problem Determination Guide and search on multiple file system manager failures). You may list which node is currently assigned as the file system manager by issuing the `mmismgr` command or change which node has been assigned to this task via the `mmchmgr` command.
The metanode

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

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

A file system consists of a set of disks (a stripe group) which are used to store:

- "Metadata"
- "Quota files" on page 49
- "Log files" on page 49
- "User data" on page 49

This set of disks is listed in a file system descriptor which is at a fixed position on each of the disks in the stripe group. In addition, the file system descriptor contains information about the state of the file system.

Metadata

Within each file system, files are written to disk as in traditional UNIX file systems, using inodes, indirect blocks, and data blocks. Inodes and indirect blocks are considered metadata, as distinguished from data, or actual file content. You can control which disks GPFS uses for storing metadata when you create disk descriptors at file system creation time.

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

A file starts out with direct pointers to data blocks in the inodes (a zero level of indirection). As the file increases in size to the point where the inode cannot hold enough direct pointers, the indirection level is increased by adding an indirect block and moving the direct pointers there. Subsequent levels of indirect blocks are added as the file grows. This allows file sizes to grow up to the largest supported file system size.
Notes:
1. The maximum number of file systems that may exist within a GPFS nodeset is 32.
2. The maximum file system size supported by IBM Service is 100TB.
3. The maximum number of files within a file system cannot exceed the architectural limit of 256 million.
4. The maximum indirection level supported by IBM Service is 3.

Using the file system descriptor to find all of the disks which make up the file system’s stripe group, and their size and order, it is possible to address any block in the file system. In particular, it is possible to find the first inode, which describes the inode file, and a small number of inodes which are the core of the rest of the file system. The inode file is a collection of fixed length records that represent a single file, directory, or link. The unit of locking is the single inode because the inode size must be a multiple of the sector size (the inode size is internally controlled by GPFS). Specifically, there are fixed inodes within the inode file for the:
- Root directory of the file system
- Block allocation map
- Inode allocation map

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

Block allocation map
The block allocation map is a collection of bits that represent the availability of disk space within the disks of the file system. One unit in the allocation map represents a subblock or 1/32 of the block size of the file system. The allocation map is broken into regions which reside on disk sector boundaries. The number of regions is set at file system creation time by the parameter that specifies how many nodes will access this file system. The regions are separately locked and, as a result, different nodes can be allocating or deallocating space represented by different regions independently and concurrently.

Inode allocation map
The inode allocation file represents the availability of inodes within the inode file. This file represents all the files, directories, and links that can be created. The mmchfs command can be used to change the maximum number of files that can be created in the file system.
Quota files
For file systems with quotas installed, quota files are created at file system creation. There are two quota files for a file system:
1. **user.quota** for users
2. **group.quota** for groups

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

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

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

Log files
Log files are created at file system creation. Additional log files may be created if needed. Log files are always replicated and are found through a pointer in the file system descriptor. The file system manager assigns a log file to each node accessing the file system.

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

There are certain failure cases where blocks are marked allocated but not part of a file, and this can be recovered by running **mmfsck** on-line or off-line. GPFS always replicates its log. There are two copies of the log for each executing node. Log recovery is run:
1. As part of the recovery of a node failure affecting the objects that the failed node might have locked.
2. As part of a **mount** after the file system has been unmounted everywhere.

User data
The remaining space is allocated from the block allocation map as needed and is used for user data and directories.

GPFS and memory
GPFS uses three areas of memory:
1. Memory allocated from the kernel heap
2. Memory allocated within the daemon segment
3. Shared segments accessed from both the daemon and the kernel

The kernel memory is used for control structures such as vnodes and related structures that establish the necessary relationship with AIX.
The file system manager node requires more daemon memory since token state for the entire file system is stored there. The daemon memory is used for structures that persist for the execution of a command or I/O operation, and also for states related to other nodes. File system manager functions use daemon storage.

Shared segments consist of both pinned and unpinned storage, which is allocated at daemon start-up. The pinned storage is labeled, pagepool and is controlled by configuration parameters. In a non-pinned area of the shared segment, GPFS keeps information about open and recently opened files. This information is held in two forms:
1. A full inode cache
2. A stat cache

The GPFS administrator controls the size of these caches through the mmconfig and mmchconfig commands.

The inode cache contains copies of inodes for open files and for some recently used files which are no longer open. The number of inodes cached is controlled by the maxFilesToCache parameter. The number of inodes for recently used files is constrained by how much the maxFilesToCache parameter exceeds the current number of open files in the system. However, you may have open files in excess of the maxFilesToCache parameter.

The stat cache contains enough information to respond to inquiries about the file and open it, but not enough information to read from it or write to it. There is sufficient data from the inode to respond to a stat( ) call (the system call under commands such as ls -l). A stat cache entry consumes about 128 bytes which is significantly less memory than a full inode. The default value is $4 \times \text{maxFilesToCache}$. This value may be changed via the maxStatCache parameter on the mmchconfig command. The stat cache entries are kept for:
1. Recently accessed files
2. Directories recently accessed by a number of stat( ) calls

GPFS will prefetch data for stat cache entries if a pattern of use indicates this will be productive. Such a pattern might be a number of ls -l commands issued for a large directory.

Note: Each entry in the inode cache and the stat cache requires appropriate tokens to ensure the cached information remains correct and the storage of these tokens on the file system manager node. Depending on the usage pattern, a degradation in performance can occur when the next update of information on another node requires that the token be revoked.

Pagepool is used for the storage of data and metadata in support of I/O operations. With some access patterns, increasing the amount of pagepool storage may increase I/O performance for file systems with the following operating characteristics:
- Heavy use of writes that can be overlapped with application execution
- Heavy reuse of files and sequential reads of a size such that prefetch will benefit the application

Component interfaces
The correct operation of GPFS is directly dependent upon:
- "Program interfaces"
- "Socket communications" on page 51

Program interfaces
The correct operation of the GPFS file system in a cluster environment depends on a number of other programs. Specifically, GPFS depends on the correct operation of:
- RSCT
Group Services, a component of RSCT, is used to notify GPFS of failures of the GPFS daemon and the components upon which it is dependent. Specifically, GPFS monitors the GPFS adapter’s Group Services group. Should Group Services not be available when the GPFS daemon is started, GPFS will exit and attempt to restart. This cycle will be repeated until Group Services becomes available. Once a connection with Group Services is established, GPFS monitors the ethernet group to ensure it is available. If the network is not available, GPFS will retry and put periodic messages into /var/adm/ras/mmfs.log.latest indicating that it is waiting for the network.

The communication path between nodes is built at the first attempt to communicate. Each node in the nodeset is required to communicate with the configuration manager and the file system manager during start-up and mount processing. The establishment of other communication paths depends upon application usage among nodes. Once a connection is established, it remains active until the GPFS daemon goes down on the nodes.

**Socket communications**

There are several component interfaces that affect GPFS behavior. These are socket communications between:

- User commands and the daemon
- Instances of daemon code

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

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

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

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

The daemon also uses sockets to communicate with other instances of the file system on other nodes. Specifically, the daemon on each node communicates with the file system manager for allocation of logs, allocation segments, and quotas, as well as for various recovery and configuration flows. GPFS requires an active internode communications path between all nodes in a nodeset for locking, metadata coordination, administration commands, and other internal functions. The existence of this path is necessary for the correct operation of GPFS. The instance of the GPFS daemon on a node will go down if it senses that this communication is not available to it. If communication is not available to another node, one of the two nodes will exit GPFS.
Application and user interaction with GPFS

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

1. “Operating system commands”
2. “Operating system calls” on page 53
3. “GPFS command processing” on page 56

Operating system commands

Operating system commands operate on GPFS data during:

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

Initialization

GPFS initialization can be done automatically as part of the node start-up sequence, or manually using the mmstartup command to start the daemon. The daemon start-up process loads the necessary kernel extensions, if they have not been previously loaded by an earlier instance of the daemon subsequent to the current IPL of this node. The initialization sequence then waits for the configuration manager to declare that a quorum exists. If Group Services reports that this node is the first to join the GPFS group, this node becomes the configuration manager. When quorum is achieved, the configuration manager changes the state of the group from initializing to active using Group Services interfaces. This transition is evident in a message to the GPFS console file (/var/adm/ras/mmfs.log.latest).

The initialization sequence also awaits membership in the Group Services adapter membership group, if not already established. Note that Group Services will queue the request to join these groups if a previous failure is still being recovered, which will delay initialization. This is crucial if the failure being recovered is a failure of this node. Completion of the group join means that all necessary failure recovery is complete.

Initializing GPFS in an AIX cluster environment: The initialization sequence also awaits membership in the GPFS adapter’s Group Services group, if not already established. Note that Group Services will queue the request to join this group if a previous failure is still being recovered, which will delay initialization. This is crucial if the failure being recovered is a failure of this node. Completion of the group join means that all necessary failure recovery is complete.

When this state change from initializing to active has occurred, the daemon is ready to accept mount requests.

mount

GPFS file systems are mounted using the mount command, which builds the structures that serve as the path to the data. GPFS mount processing is performed on both the node requesting the mount and the file system manager node. If there is no file system manager, a call is made to the configuration manager, which appoints one. The file system manager will ensure that the file system is ready to be mounted. This includes checking that there are no conflicting utilities being run by mmfsck or mmcheckquota, for example, and running any necessary log processing to ensure that metadata on the file system is consistent.

On the local node the control structures required for a mounted file system are initialized and the token management function domains are created. In addition, paths to each of the disks which make up the file system are opened. Part of mount processing involves unfencing the disks, which may be necessary if this node had previously failed. This is done automatically without user intervention except in the rare case of a two-node nodeset using single-node quorum (see the General Parallel File System for AIX 5L: AIX Clusters Problem Determination Guide and search on single-node quorum). If insufficient disks are up, the
mount will fail. That is, in a replicated system if two disks are down in different failure groups, the mount will fail. In a non-replicated system, one disk down will cause the mount to fail.

**Note:** There is a maximum of 32 file systems that may exist within a GPFS nodeset.

### Operating system calls

The most common interface is through normal file system calls to the operating system which are relayed to GPFS if data in a GPFS file system is involved. This uses GPFS code in a kernel extension which attempts to satisfy the application request using data already in memory. If this can be accomplished, control is returned to the application through the operating system interface. If the request requires resources that are not available at the time, the request is transferred for execution by a daemon thread. The daemon threads wait for work in a system call in the kernel, and are scheduled as necessary. Services available at the daemon level are the acquisition of tokens and disk I/O.

Operating system calls operate on GPFS data during:

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

**open**

The *open* of a GPFS file involves the application making a call to the operating system specifying the name of the file. Processing of an *open* involves two stages:

1. The directory processing required to identify the file specified by the application.
2. The building of the required data structures based on the inode.

The kernel extension code will process the directory search for those directories which reside in GPFS (part of the path to the file may be directories in other physical file systems). If the required information is not in memory, the daemon will be called to acquire the necessary tokens for the directory or part of the directory needed to resolve the lookup. It will also read the directory entry into memory.

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

- If no other node has this file open, this node becomes the metanode
- If another node has a previous open, then that node is the metanode and this node will interface with the metanode for certain parallel write situations
- If the *open* involves creation of a new file, the appropriate locks are obtained on the parent directory and the inode allocation file block. The directory entry is created, an inode is selected and initialized and then *open* processing is completed.

**read**

The GPFS *read* function is invoked in response to a *read* system call and a call through the operating system vnode interface to GPFS. *read* processing falls into three levels of complexity based on system activity and status:

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

*Buffer and locks available in memory:* The simplest *read* operation occurs when the data is already available in memory, either because it has been prefetched or because it has been read recently by another *read* call. In either case, the buffer is locally locked and the data is copied to the application data
area. The lock is released when the copy is complete. Note that no token communication is required
because possession of the buffer implies that we at least have a read token that includes the buffer. After
the copying, prefetch is initiated if appropriate.

**Tokens available locally but data must be read:** The second, more complex, type of **read** operation is
necessary when the data is not in memory. This occurs under three conditions:
1. The token has been acquired on a previous **read** that found no contention.
2. The buffer has been stolen for other uses.
3. On some random **read** operations.
   In the first of a series of random reads the token will not be available locally, but in the second read it
   might be available.

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

**Data and tokens must be acquired:** The third, and most complex **read** operation requires that tokens
as well as data be acquired on the application node. The kernel code determines that the data is not
available locally and sends the message to the daemon waiting after posting the message. The daemon
thread determines that it does not have the required tokens to perform the operation. In that case, a token
acquire request is sent to the token management server. The requested token specifies a required length
of that range of the file, which is needed for this buffer. If the file is being accessed sequentially, a desired
range of data, starting at this point of this read and extending to the end of the file, is specified. In the
event that no conflicts exist, the desired range will be granted, eliminating the need for **token calls** on
subsequent reads. After the minimum token needed is acquired, the flow proceeds as in the token
management function on page 46.

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

**write**

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

Like many open systems file systems, GPFS moves data from a user buffer into a file system buffer
synchronously with the application **write** call, but defers the actual write to disk. This technique allows
better scheduling of the disk and improved performance. The file system buffers come from the memory
allocated by the **pagepool** parameter in the **mmconfig** or **mmchconfig** command. Increasing this value
may allow more writes to be deferred, which improves performance in certain workloads.

A block of data is scheduled to be written to a disk when:
- The application has specified synchronous **write**.
- The system needs the storage.
- A token has been revoked.
- The last byte of a block of a file being written sequentially is written.
- A **sync** is done.

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

**write** processing falls into three levels of complexity based on system activity and status:
1. Buffer available in memory
2. Tokens available locally but data must be read
3. Data and tokens must be acquired

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

**Buffer available in memory:** The simplest path involves a case where a buffer already exists for this block of the file but may not have a strong enough token. This occurs if a previous write call accessed the block and it is still resident in memory. The write token already exists from the prior call. In this case, the data is copied from the application buffer to the GPFS buffer. If this is a sequential write and the last byte has been written, an asynchronous message is sent to the daemon to schedule the buffer for writing to disk. This operation occurs on the daemon thread overlapped with the execution of the application.

**Token available locally but data must be read:** There are two situations in which the token may exist but the buffer does not:
1. The buffer has been recently stolen to satisfy other needs for buffer space.
2. A previous write obtained a desired range token for more than it needed.

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

**Data and tokens must be acquired:** The third, and most complex path through write occurs when neither the buffer nor the token exists at the local node. Prior to the allocation of a buffer, a token is acquired for the area of the file which is needed. As was true for read, if sequential operations are occurring, a token covering a larger range than is needed will be obtained if no conflicts exist. If necessary, the token management function will revoke the needed token from another node holding the token. Having acquired and locked the necessary token, the write will continue as in "Token available locally but data must be read".

**stat**
The stat( ) system call returns data on the size and parameters associated with a file. The call is issued by the ls -l command and other similar functions. The data required to satisfy the stat( ) system call is contained in the inode. GPFS processing of the stat( ) system call differs from other file systems in that it supports handling of stat( ) calls on all nodes without funneling the calls through a server.

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

Inodes are cached within GPFS in two forms:
1. Full inode
2. Limited stat cache form

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

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

### GPFS command processing

GPFS commands fall into two categories: those that are processed locally and those that are processed at the file system manager for the file system involved in the command. The file system manager is used to process any command that alters the state of the file system. When commands are issued but the file system is not mounted, a file system manager is appointed for the task. The `mmchdisk` command and the `mmfsck` command represent two typical types of commands which are processed at the file system manager.

#### The mmchdisk command

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

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

All of these functions operate in a mode that is mutually exclusive with other commands that change the state of the file system. However, they coexist with commands that perform typical file system operations. Thus, the processing flows as follows:

1. Command processing occurs on the local node, which forwards the command to the file system manager. Status reports and error messages are shipped back through the command processor and reported via standard output and standard error.
2. The file system manager reads all metadata looking for replicated data or metadata that has a copy on the failed disks. When a block is found, the later copy is moved over the copy on the recovering disk. If required metadata cannot be read, the command exits leaving the disks with an availability of `recovering`. This can occur because the `mmchdisk` command did not specify all `down` disks or because additional disks have failed during the operation of the command.
3. When all required data has been copied to the recovering disk, the disk is marked `up` and is automatically used by the file system, if mounted.

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

#### The mmfsck Command

The `mmfsck` command is a traditional UNIX command that repairs file system structures. `mmfsck` operates in two modes:

1. on-line
2. off-line

For performance reasons, GPFS logging allows the condition where disk blocks are marked `used` but not actually part of a file after a node failure. The on-line version of `mmfsck` cleans up that condition. Running
The on-line version of `mmfsck` runs on the file system manager and scans all inodes and indirect blocks looking for disk blocks which are allocated but not used. If authorized to repair the file system, it releases the blocks. If not authorized to repair the file system, it reports the condition to standard output on the invoking node.

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

### Recovery

In order to understand the GPFS recovery process, you need to be familiar with Group Services. In particular, it should be noted that only one state change, such as the loss or initialization of a node, can be processed at a time and subsequent changes will be queued. This means that the entire failure processing must complete before the failed node can join the group again. Group Services also processes all failures first, which means that GPFS will handle all failures prior to completing any recovery.

GPFS uses two groups to process failures of nodes, or GPFS failure on other nodes. The primary group is used to process failures. The secondary group is used to restart a failure protocol if a second failure is reported. This may be an actual second failure, or one that occurred at the same time as the first but was detected by Group Services later. The only function of the second group is to abort the current protocol and restart a new one processing all known failures.

GPFS recovers from node failure using notifications provided by Group Services. When notified that a node has failed or that the GPFS daemon has failed on a node, GPFS invokes recovery for each of the file systems that were mounted on the failed node. If necessary, a new Configuration Manager is selected prior to the start of actual recovery, or new file system managers are selected for any file systems that no longer have one, or both. This processing occurs as the first phase of recovery and occurs on the configuration manager. This processing must complete before other processing can be attempted and is enforced using Group Services barriers.

The file system manager for each file system fences the failed node from the disks comprising the file system. If the file system manager is newly appointed as a result of this failure, it rebuilds token state by querying the other nodes of the group. This file system manager recovery phase is also protected by a Group Services barrier. After this is complete, the actual recovery of the log of the failed node proceeds. This recovery will rebuild the metadata that was being modified at the time of the failure to a consistent state with the possible exception that blocks may be allocated that are not part of any file and are effectively lost until `mmfsck` is run, on-line or off-line. After log recovery is complete, the locks held by the failed nodes are released for this file system. Completion of this activity for all file systems completes the failure processing. The completion of the protocol allows a failed node to rejoin the cluster. GPFS will unfence the failed node after it has rejoined the group.

### GPFS cluster data

GPFS commands save configuration and file system information in one or more files collectively known as GPFS cluster data. These files are not intended to be modified manually. The GPFS administration commands are designed to keep these file synchronized between each other and with the GPFS system files on each node in the nodeset. The GPFS commands constantly update the GPFS cluster data and any user modification made to this information may be lost without warning. This includes the GPFS file system stanzas in `/etc/filesystems`. 
The GPFS cluster data information is stored in the file `/var/mmfs/gen/mmsdrfs`. This file is stored on the nodes designated as the primary GPFS cluster data server and, if specified, the secondary GPFS cluster data server (see “GPFS cluster data servers” on page 14).

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

`/etc/cluster.nodes`  
Contains a list of all nodes that belong to the local nodeset.

`/etc/filesystems`  
Contains lists for all GPFS file systems that exist in the nodeset.

`/var/mmfs/gen/mmsdrfs`  
Contains a local copy of the `mmsdrfs` file found on the primary and secondary GPFS cluster data server nodes.

`/var/mmfs/etc/mmfs.cfg`  
Contains GPFS daemon startup parameters.

`/var/mmfs/etc/cluster.preferences`  
Contains a list of the nodes designated as file system manager nodes.

The master copy of all GPFS configuration information is kept in the file `mmsdrfs` on the primary GPFS cluster data server node. The layout of this file is defined in `/usr/lpp/mmfs/bin/mmsdrsddef`. The first record in the `mmsdrfs` file contains a generation number. Whenever a GPFS command causes something to change in any of the nodesets or any of the file systems, this change is reflected in the `mmsdrfs` file and the generation number is incremented. The latest generation number is always recorded in the `mmsdrfs` file on the primary and secondary GPFS cluster data server nodes.

When running GPFS administration commands in a GPFS cluster, it is necessary for the GPFS cluster data to be accessible to the node running the command. Commands that update the `mmsdrfs` file require that both the primary and secondary GPFS cluster data server nodes are accessible. Similarly, when the GPFS daemon starts up, at least one of the two server nodes must be accessible.
Appendix B. Considerations for GPFS applications

Application design should take into consideration:

- Exceptions to Open Group technical standards
- Application support

Exceptions to Open Group technical standards

GPFS is designed so that most applications written to The Open Group technical standard for file system calls can access GPFS data with no modification, however, there are some exceptions.

Applications that depend on exact reporting of changes to the following fields returned by the stat() call may not work as expected:

1. mtime
2. ctime
3. atime

Providing exact support for these fields would require significant performance degradation to all applications executing on the system. These fields are guaranteed accurate when the file is closed.

Alternatively, you may:

- Use the GPFS calls, gpfs_stat() and gpfs_fstat() to return exact mtime and atime values.
- Choose to accept some performance degradation and use the mmcrfs or mmchfs command with the -E yes option to report on exact mtime values.

The delayed update of the information returned by the stat() call also impacts system commands which display disk usage, such as du or df. The data reported by such commands may not reflect changes that have occurred since the last sync of the file system. For a parallel file system, a sync does not occur until all nodes have individually synchronized their data. On a system with no activity, the correct values will be displayed after the sync daemon has run on all nodes.

Application support

Applications access GPFS data through the use of standard AIX 5L system calls and libraries. Support for larger files is provided through the use of AIX 5L 64-bit forms of these libraries. See the AIX 5L product documentation at www.ibm.com/servers/aix/library/techpubs.html for details.
Appendix C. Restrictions and conventions for GPFS

This appendix lists by activity, usage restrictions and conventions which should be followed when using GPFS (for restrictions regarding the use of DMAPI, see the General Parallel File System for AIX 5L: AIX Clusters Data Management API Guide) and assumes you are familiar with the GPFS product:

- “GPFS cluster configuration”
- “GPFS nodeset configuration” on page 62
- “Starting GPFS” on page 62
- “GPFS file system configuration” on page 63
- “GPFS cluster administration” on page 63
- “GPFS nodeset administration” on page 64
- “GPFS file system administration” on page 64
- “Disk administration in your GPFS file system” on page 66
- “Communicating file accessing patterns” on page 67
- “System configuration” on page 68

GPFS cluster configuration

These restrictions apply to the creation of your GPFS cluster:

1. The only valid GPFS cluster types are rpd and hacmp.
2. A node may only belong to one GPFS cluster at a time.
3. The hostname or IP address used for a node must refer to the communications adapter over which the GPFS daemons communicate. Alias interfaces are not allowed. Use the original address or a name that is resolved by the host command to that original address. You may specify a node using any of these forms:

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short hostname</td>
<td>k145n01</td>
</tr>
<tr>
<td>Long hostname</td>
<td>k145n01.kgn.ibm.com</td>
</tr>
<tr>
<td>IP address</td>
<td>9.119.19.102</td>
</tr>
</tbody>
</table>

4. For a cluster type of hacmp, any node to be included in your GPFS cluster must be a properly configured node in an existing HACMP cluster.

For further information, see the High Availability Cluster Multi-Processing for AIX: Enhanced Scalability Installation and Administration Guide.

5. For a cluster type of rpd, any node to be included in your GPFS cluster must be a properly configured node in an existing RSCT peer domain.

For further information, see the Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference.

6. Nodes specified in the NodeFile which are not available when the mmcrcluster command is issued must be added to the cluster by issuing the mmaddcluster command.

7. You must have root authority to run the mmcrcluster command.

8. The mmcrcluster command will only be successful if the primary server and, if specified, the secondary server are available.

9. The authentication method between nodes in the GPFS cluster must be established when the mmcrcluster command is issued:

   a. When using rcp and rsh for remote communication, a properly configured /.rhosts file must exist in the root users home directory on each node in the GPFS cluster.

© Copyright IBM Corp. 2002
b. If you have designated the use of a different remote communication program on either the
\texttt{mmcrcluster} or the \texttt{mmchcluster} command, you must ensure:

1) Proper authorization is granted to all nodes in the GPFS cluster.
2) The nodes in the GPFS cluster can communicate without the use of a password.

The remote copy and remote shell command must adhere to the same syntax form as \texttt{rcp} and \texttt{rsh} but may implement an alternate authentication mechanism.

\section*{GPFS nodeset configuration}

These restrictions apply to the configuration of your GPFS nodeset:

1. You may not configure a GPFS nodeset until you have created your GPFS cluster.
2. The hostname or IP address used for a node must refer to the communications adapter over which the GPFS daemons communicate. Alias interfaces are not allowed. Use the original address or a name that is resolved by the \texttt{host} command to that original address. You may specify a node using any of these forms:

<table>
<thead>
<tr>
<th>Format</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short hostname</td>
<td>k145n01</td>
</tr>
<tr>
<td>Long hostname</td>
<td>k145n01.kgn.ibm.com</td>
</tr>
<tr>
<td>IP address</td>
<td>9.119.19.102</td>
</tr>
</tbody>
</table>

3. A node may belong to only one GPFS nodeset at a time.
4. If the disks in your system are purely Fibre Channel, the maximum supported number of nodes in a GPFS nodeset is 32.
5. If the disks in your system are SSA or a combination of SSA and Fibre Channel, the maximum supported number of nodes in a GPFS nodeset is eight.
6. A nodeset identifier can be at most eight alphanumeric characters, including the underscore character. The identifier cannot be a reserved word such as \texttt{AVAIL}, \texttt{vsd}, \texttt{rpdp}, \texttt{hacmp}, or \texttt{lc} and it cannot be the number zero. The nodeset identifier cannot be changed once it is set.
7. Before creating a GPFS nodeset you must first verify that all of the nodes to be included in the nodeset are members of the GPFS cluster (see the \texttt{mmlscluster} command).
8. All nodes in a GPFS nodeset must belong to the same GPFS cluster.
9. The combined amount of memory to hold inodes, control data structures, and the stat cache is limited to 50% of the physical memory.

\section*{Starting GPFS}

These restrictions apply to starting GPFS:

1. \textit{DO NOT} start GPFS until it is configured.
2. Quorum must be met in order to successfully start GPFS.
3. You must have root authority to issue the \texttt{mmstartup} command.
4. When using \texttt{rcp} and \texttt{rsh} for remote communication, a properly configured \texttt{/rhosts} file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the \texttt{mmcrcluster} or the \texttt{mmchcluster} command, you must ensure:
   a. Proper authorization is granted to all nodes in the GPFS cluster.
   b. The nodes in the GPFS cluster can communicate without the use of a password.
5. You may issue the \texttt{mmstartup} command from any node in the GPFS cluster.
GPFS file system configuration

These restrictions apply to configuring your GPFS file system:
1. A GPFS file system may only be accessed from a single nodeset.
2. Your logical volumes must be created via the `mmcrlv` command prior to creating your file system.
3. GPFS may not be used for any file systems that are required by AIX to be in the `rootvg`.
4. File system names must be unique across GPFS nodesets and cannot be an existing entry in `/dev`.
5. The maximum number of file systems supported is 32.
6. The maximum GPFS file system size that can be mounted is limited by the control structures in memory required to maintain the file system. These control structures, and consequently the maximum mounted file system size, are a function of the block size of the file system.
   - If your file systems have a 16 KB block size, you may have one or more file systems with a total size of 1 TB mounted.
   - If your file systems have a 64 KB block size, you may have one or more file systems with a total size of 10 TB mounted.
   - If your file systems have a 256 KB or greater block size, you may have file systems mounted with a total size of not greater than 200 TB where no single file system exceeds 100 TB.
7. The maximum level of indirection is 3.
8. The maximum number of files within a file system cannot exceed the architectural limit of 256 million.
9. The maximum number of disks in a GPFS file system is 1024.
   The actual number of disks in your file system may be constrained by products other than GPFS which you have installed. Refer to your individual product documentation.
10. The maximum value for `pagepool` is 512 MB per node.
11. The maximum block size supported is 1024 KB.
   If you choose a block size larger than 256 KB (the default), you must run `mmchconfig` to change the value of `maxblocksize` to a value at least as large as `BlockSize`.
12. The maximum replication value for both data and metadata is 2.
13. The value for `BlockSize` cannot be changed without recreating the file system.
14. The value for `NumNodes` cannot be changed after the file system has been created.
15. If the `mmcrfs` command is interrupted for any reason, you must use the `-v no` option on the next invocation of the command.
16. The `mmconfig` command may only be run once. Any changes to your GPFS configuration after the command has been issued, must be made by using the `mmchconfig` command.
17. When changing both `maxblocksize` and `pagepool`, these conventions must be followed or the command will fail:
   - When increasing the values, `pagepool` must be specified first.
   - When decreasing the values, `maxblocksize` must be specified first.
18. All shared disks or disk arrays must be directly attached to all nodes in the nodeset.
19. The largest disk size supported is 1 TB.

GPFS cluster administration

These restrictions apply to administering your GPFS cluster:
1. You must have root authority to run the `mmaddcluster`, `mmdelcluster`, `mmchcluster`, and `mmlscluster` commands.
2. A node may only belong to one GPFS cluster at a time.
3. When adding a node to a GPFS cluster, it must be available for the `mmaddcluster` command to be successful.
4. The PrimaryServer and, if specified, the SecondaryServer must be available for the mmaddcluster, mmdelcluster, and mmiscluster commands to be successful.

5. The mmchcluster command, when issued with either the -p or -s option, is designed to operate in an environment where the current PrimaryServer and, if specified, the SecondaryServer are not available. When specified with any other options, the servers must be available for the command to be successful.

6. A node being deleted cannot be the primary or secondary GPFS cluster data server unless you intend to delete the entire cluster. Verify this by issuing the mmiscluster command. If a node to be deleted is one of the servers and you intend to keep the cluster, issue the mmchcluster command to assign another node as the server before deleting the node.

GPFS nodeset administration

These restrictions apply to administering your GPFS nodeset:

1. The nodes being added to the nodeset must belong to the GPFS cluster. Issue the mmiscluster command to display the available nodes or add nodes to the cluster by issuing the mmaddcluster command.

2. Before you can delete a node, you must issue the mmshutdown command to unmount all of the GPFS file systems and stop GPFS on the node to be deleted.

3. When a node is deleted from a GPFS nodeset, its entry is not automatically deleted from the nodeset configuration. Instead the node is only marked as deleted. This allows nodes to be deleted without having to stop GPFS on all nodes. Such deleted nodes are not a factor when calculating quorum. They are also available to the mmaddnode and mmconfig commands for inclusion into another GPFS nodeset. If you want to remove any deleted node entries from the nodeset configuration, you must use the -c option on the mmdelnodel command. The GPFS daemon must be stopped on all of the nodes in the nodeset, not just the ones being deleted. This can be done when the node is deleted or anytime later.

4. If single-node quorum is enabled, nodes cannot be added to or deleted from a nodeset without stopping the GPFS daemon on both nodes.

GPFS file system administration

The following restrictions apply to file system administration:

1. Root authority is required to perform all GPFS administration tasks except those with a function limited to listing GPFS operating characteristics or modifying individual user file attributes.

2. In order to use new function (see "What's new" on page xi), you must change the file system format by issuing the mmchfs command with the -V option.

3. The maximum GPFS file system size that can be mounted is limited by the control structures in memory required to maintain the file system. These control structures, and consequently the maximum mounted file system size, are a function of the block size of the file system.

   • If your file systems have a 16 KB block size, you may have one or more file systems with a total size of 1 TB mounted.
   • If your file systems have a 64 KB block size, you may have one or more file systems with a total size of 10 TB mounted.
   • If your file systems have a 256 KB or greater block size, you may have file systems mounted with a total size of not greater than 200 TB where no single file system exceeds 100 TB.

4. You must create logical volumes for use with your file system via the mmcrlv command. The mmcrlv command will:

   • Use SCSI-3 persistent reserve on disks which support it or SSA fencing if that is supported by the disk. Otherwise disk leasing will be used. See General Parallel File System for AIX 5L: AIX Clusters Concepts, Planning, and Installation Guide and search on disk fencing.
• Have bad-block relocation automatically turned off. Accessing disks concurrently from multiple systems using lvm bad-block relocations could potentially cause conflicting assignments. As a result, software bad-block relocation is turned off allowing the hardware bad-block relocation supplied by your disk vendor to provide protection against disk media errors.

When creating a logical volume, you must have write access to where the disk descriptor file is located.

5. When using rcp and rsh for remote communication, a properly configured .rhosts file must exist in the root user’s home directory on each node in the GPFS cluster. If you have designated the use of a different remote communication program on either the mmcrecluster or the mmchcluster command, you must ensure:
   a. Proper authorization is granted to all nodes in the GPFS cluster.
   b. The nodes in the GPFS cluster can communicate without the use of a password.

   The remote copy and remote shell command must adhere to the same syntax form as rcp and rsh but may implement an alternate authentication mechanism.

6. In order to run mmfsck off-line to repair a file system, you must unmount your file system.

7. When replacing quota files with either the -u or the -g option on the mmcheckquota command:
   • The quota files must be in the root directory of the file system.
   • The file system must be unmounted.

8. Multi-node quorum must be maintained when adding or deleting nodes from your GPFS nodeset.

9. You must unmount the file system on all nodes before deleting it.

10. You must unmount a file system on all nodes before moving it to a different nodeset.

11. When issuing mmchfs to enable DMAPI, the file system cannot be in use.

Commands may be run from various locations within your system configuration. Use this information to ensure the command is being issued from an appropriate location and is using the correct syntax (see the individual commands for specific rules regarding the use of that command):

1. Commands which may be issued from any node in the GPFS cluster running GPFS:

   Note: If the command is intended to run on a nodeset other than the one you are on, you must specify the nodeset using the -C option.

   • mmaddnode
   • mmchconfig
   • mmcrfs
   • mmstartup
   • mmshutdown

2. Commands which require that Device be the first operand and may be issued from any node in the GPFS cluster running GPFS

   • mmadddisk
   • mmchdisk
   • mmchfs
   • mmchmgr
   • mmdefragfs
   • mmdeldisk
   • mmdelfs
   • mmdf
   • mmfsck
   • mmlsdisk
Either Device or NodesetId must be specified.

3. Commands which require GPFS to be running on the node from which the command is issued:
   - mmcheckquota
   - mmdefedquota
   - mmdefquotaoof
   - mmdefquotaon
   - mmedquota
   - mmlsquota
   - mmquotaoff
   - mmquotaon
   - mmrepquota

4. Commands which require the file system be mounted on the GPFS nodeset from which the command is issued:
   - mmchattr
   - mmdelacl
   - mmeditacl
   - mmgetacl
   - mmlsatrr
   - mmputacl

5. Commands which may be issued from any node in the GPFS cluster where GPFS is installed:
   - mmaddcluster
   - mmchcluster
   - mmconfig
   - mmcrcluster
   - mmclv
   - mmdecluster
   - mmdelv
   - mmdelnode
   - mmliscluster
   - mmlisconfig
   - mmlsgpfsdisk
   - mmlsnode
   - mmstartup

Disk administration in your GPFS file system

These restrictions apply to administering the disks in your GPFS file system:
1. The maximum number of disks in a GPFS file system is 1024.
   The actual number of disks in your file system may be constrained by products other than GPFS which you have installed. Refer to your individual product documentation.
2. The largest disk size supported is 1 TB.
3. You cannot run `mmfsck` on a file system that has disks in a **down** state.

4. A disk remains suspended until it is explicitly resumed. Restarting GPFS or rebooting the nodes does not restore normal access to a suspended disk.

5. A disk remains down until it is explicitly started. Restarting GPFS or rebooting the nodes does not restore normal access to a down disk.

6. Only logical volumes created by the `mmcrlv` command may be used. This ensures:
   a. GPFS will exploit SCSI-3 persistent reserve if the disk supports it.
   b. Bad-block relocation is automatically turned off. Accessing disks concurrently from multiple systems using lvm bad-block relocations could potentially cause conflicting assignments. As a result, turning off software bad-block relocation allows the hardware bad-block relocation supplied by your disk vendor to provide protection against disk media errors. Bad-block relocation is automatically turned off for logical volumes created via the command.

7. When creating a logical volume by issuing the `mmcrlv` command, you must have write access to the disk descriptor file.

8. When referencing a disk, you must use the logical volume name.

9. All disks or disk arrays must be directly attached to all nodes in the nodeset.

10. You cannot protect your file system against disk failure by mirroring data at the LVM level. You must use replication or RAID devices to protect your data (see **Recoverability considerations**).

11. Single-node quorum is only supported when disk leasing is not in effect. **Disk leasing** is activated if any disk in any filesystem in the nodeset is not using SSA fencing or SCSI-3 persistent reserve.

12. Before deleting a disk use the `mmdf` command to determine whether there is enough free space on the remaining disks to store the file system.

13. Disk accounting is not provided at the present time.

14. After migrating to a new level of GPFS, before you can use an existing logical volume, which was not part of any GPFS file system at the time of migration, you must:
   a. Export the logical volume
   b. Recreate the logical volume
   c. Add the logical volume to a file system

**Communicating file accessing patterns**

These restrictions apply when using the `gpfs_fcntl()` library calls:

1. The value of the total length of the header data structure, `gpfsFcntlHeader_t`, cannot exceed the value of `GPFS_MAX_FCNTL_LENGTH` as defined in the header file, `gpfs_fcntl.h`. The current value of `GPFS_MAX_FCNTL_LENGTH` is 64K bytes.

2. The value of the `fcntlReserved` field of the header data structure, `gpfsFcntlHeader_t`, must be set to zero.

3. The value of the `fcntlVersion` field of the header data structure `gpfsFcntlHeader_t` must be set to the current version number of the `gpfs_fcntl()` library call, as defined by `GPFS_FCNTL_CURRENT_VERSION` in the header file `gpfs_fcntl.h`. The current version number is one.

4. For the `gpfsMultipleAccessRange_t` hint, up to `GPFS_MAX_RANGE_COUNT`, as defined in the header file `gpfs_fcntl.h`, blocks may be given in one multiple access range hint. The current value of `GPFS_MAX_RANGE_COUNT` is eight. Depending on the current load, GPFS may initiate prefetching of some or all of the blocks.

5. The `gpfsCancelHints_t` hint may only cancel the `gpfsMultipleAccessRange_t` hint. This directive may not cancel other directives.

6. Because an application-level `read` or `write` may be split across several agents, Posix `read` and `write` file atomicity is not enforced while in data shipping mode.
7. A file in data shipping mode cannot be written through any file handle that was not associated with the data shipping collective through a `gpfsDataShipStart_t` directive.

8. Calls that are not allowed on a file that has data shipping enabled:
   - `chac1`
   - `fchac1`
   - `chmod`
   - `fchmod`
   - `chown`
   - `fchown`
   - `chownx`
   - `fchownx`
   - `link`

9. The `gpfsDataShipStart_t` directive can only be cancelled by a `gpfsDataShipStop_t` directive.

10. For the `gpfsDataShipMap_t` directive, the value of `partitionSize` must be a multiple of the number of bytes in a single file system block.

---

**System configuration**

GPFS requires invariant network connections. The port on a particular IP address must be a fixed piece of hardware that is translated to a fixed network adapter and is monitored for failure. Topology Services should be configured to heartbeat over this invariant address. In an HACMP environment, see the [High Availability Cluster Multi-Processing for AIX: Enhanced Scalability Installation and Administration Guide](#) and search on *The Topology Services Subsystem*. In an RSCT peer domain environment, see the [Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference](#) and search on *The Topology Services Subsystem*. 

---

68 GPFS AIX Clusters Concepts, Planning, and Installation Guide
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 IBM’s product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM’s intellectual property rights 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 give 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 10594-1785
USA

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, 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.

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
Intellectual Property Law
2455 South Road,P386

© Copyright IBM Corp. 2002
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.

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 illustrates 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. 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 IBM’s application programming interfaces.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States or other countries or both:

- AFS
- AIX
- AIX 5L
- @server
- Enterprise Storage Server
- IBM
- IBMLink
- Netfinity
- pSeries
- SP
- TotalStorage
- xSeries

The Open Group is a trademark of The Open Group.

Linux is a registered trademark of Linus Torvalds.

Network File System is a trademark of Sun Microsystems, Inc.

NFS is a registered trademark of Sun Microsystems, Inc.
ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

Sun is a trademark of Sun Microsystems, Inc.

UNIX is a registered trademark of the Open Group in the United States and other countries.

Other company, product, and service names may be the trademarks or service marks of others.
AIX cluster environment. The AIX cluster environment is based on the use of either the RSCT subsystem of AIX 5L (GPFS cluster type rpd) or the HACMP/ES program product (GPFS cluster type hacmp).

block utilization. The measurement of the percentage of used subblocks per allocated blocks.

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 "GPFS cluster" on page 74).

configuration manager. The GPFS node that selects file system managers and determines whether quorum exists. The oldest continuously operating node in the file system group as monitored by Group Services, is automatically assigned as the configuration manager.

control data structures. Data structures needed to manage file data and metadata cached in memory. This includes hash tables and link pointers for finding cached data, lock states and tokens to implement distributed locking, as well as various flags and sequence numbers to keep track of updates to the cached data.

Data Management API. 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.

disk descriptor. A disk descriptor defines how a disk is to be used within a GPFS file system. Each descriptor must be in the form (second and third fields reserved):

```
DiskName:::DiskUsage:FailureGroup
```

Where DiskName is the name of the disk. This must be the name of the logical volume name. DiskUsage tells GPFS whether data, metadata, or both are to be stored on the disk. The FailureGroup designation indicates to GPFS where not to place replicas of data and metadata. All disks with a common point of failure should belong to the same failure group. Since GPFS does not place replicated information on disks in the same failure group, the availability of information is ensured even in the event of disk failure.

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. Disk leasing is a capability of the GPFS program product to interface with storage devices. Specifically, disk leasing provides control of access from multiple host systems which is useful in recovery situations. To access a storage device which is configured to use disk leasing, a host must register using a valid lease. In the event of a perceived failure, another host system may preempt that access using that valid lease which will result in the storage device not honoring attempts to read or write data on the device until the pre-empted system has re-registered. Software conventions exist in GPFS which only allow a pre-empted system to re-register after the recovery situation has been addressed. Disk leasing is activated if any disk in the file system is not using SSA fencing or SCSI-3 persistent reserve. Contrast with "persistent reserve" on page 75.

domain. (1) A set of network resources (such as applications and printers, for example) for a group of users. A user logs in to the domain to gain access to the resources, which could be located on a number of different servers in the network. (2) A group of server and client machines that exist in the same security structure. (3) A group of computers and devices on a network that are administered as a unit with common rules and procedures. Within the Internet, a domain is defined by its Internet Protocol (IP) address. All devices that share a common part of the IP address are said to be in the same domain.

event. A message from a file operation to a data management application about the action being performed on the file or file system. There are several types of events, each used for a different type of action. The event is delivered to a session according to the event disposition.

failover. The assuming of server responsibilities by the node designated as backup server, when the primary server fails.

failure group. A collection of disks that share common access paths or adaptor connection, and could all become unavailable through a single hardware failure.
file system manager. There is one file system manager per file system, which provides the following services for all the nodes using the file system:

1. Processes changes to the state or description of the file system. These include:
   - Adding disks
   - Changing disk availability
   - Repairing the file system

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

3. Controls token management.

4. Controls quota management.

fragment. The space allocated for an amount of data (usually at the end of a file) too small to require a full block, consisting of one or more subblocks (one thirty-second of block size).

G

GPFS cluster. A subset of existing cluster nodes defined as being available for use by GPFS file systems. The GPFS cluster is created via the `mmcrcluster` command. GPFS nodesets and file systems are subsequently created after the `mmcrcluster` command has been issued.

GPFS cluster data. The GPFS configuration data, which is stored on the primary and secondary GPFS cluster data servers as defined on the `mmcrcluster` command.

GPFS portability layer. The interface to the GPFS for Linux proprietary code is an open source module which each installation must build for its specific hardware platform and Linux distribution. See [www.ibm.com/servers/eserver/clusters/software/](http://www.ibm.com/servers/eserver/clusters/software/).

H

HACMP environment. The operation of GPFS based on the High Availability Cluster Multi-Processing for AIX/Enhanced Scalability (HACMP/ES) program product. This environment is defined on the `mmcrcluster` command by specifying a cluster type of `hacmp`.

I

IBM Virtual Shared Disk. The component of PSSP that allows application programs executing on different nodes access a raw logical volume as if it were local at each node. In actuality, the logical volume is local at only one of the nodes (the server node).

inode. The internal structure that describes an individual file. An inode contains file size and update information, as well as the addresses of data blocks, or in the case of large files, indirect blocks that, in turn, point to data blocks. One inode is required for each file.

J

journaled file system (JFS). The local file system within a single instance of AIX.

K

Kernel Low-Level Application Programming Interface (KLAPI). KLAPI provides reliable transport services to kernel subsystems that have communication over the SP Switch.

L

logical volume. A collection of physical partitions organized into logical partitions all contained in a single volume group. Logical volumes are expandable and can span several physical volumes in a volume group.

Logical Volume Manager (LVM). Manages disk space at a logical level. It controls fixed-disk resources by mapping data between logical and physical storage, allowing data to be discontiguous, span multiple disks, replicated, and dynamically expanded.

loose cluster environment. The operation of GPFS based on the Linux operating system. This environment is defined on the `mmcrcluster` command by specifying a cluster type of `lc`.

M

management domain. A set of nodes configured for manageablebility by the Clusters Systems Management (CSM) product. Such a domain has a management server that is used to administer a number of managed nodes. Only management servers have knowledge of the whole domain. Managed nodes only know about the servers managing them; they know nothing of each other. Contrast with "peer domain" on page 75.

metadata. Data structures that contain access information about file data. These might include inodes, indirect blocks, and directories. These data structures are used by GPFS but are not accessible to user applications.

metanode. There is one metanode per open file. The metanode is responsible for maintaining file metadata integrity. In almost all cases, the node that has had the file open for the longest period of continuous time is the metanode.

mirroring. The creation of a mirror image of data to be preserved in the event of disk failure.
**multi-node quorum.** The type of algorithm used for GPFS nodesets of 3 nodes or more. This is defined as one plus half of the number of nodes in the GPFS nodeset.

**multi-tailing.** Connecting a disk to multiple nodes.

**N**

Network File System (NFS). A distributed file system that allows users to access files and directories located on remote computers and treat those files and directories as if they were local. NFS allows different systems (UNIX or non-UNIX), different architectures, or vendors connected to the same network, to access remote files in a LAN environment as though they were local files.

**node descriptor.** A node descriptor defines how a node is to be used within GPFS.

In a Linux environment, each descriptor for a GPFS cluster must be in the form:

```
primaryNetworkNodeName::secondaryNetworkNodeName
```

`primaryNetworkNodeName`

The host name of the node on the primary network for GPFS daemon to daemon communication.

`designation`

Currently unused and specified by the double colon `::`

`secondaryNetworkNodeName`

The host name of the node on the secondary network, if one exists.

You may configure a secondary network node name in order to prevent the node from appearing to have gone down when the network is merely saturated. During times of excessive network traffic if a second network is not specified, there is the potential for the RSCT component to be unable to communicate with the node over the primary network. RSCT would perceive the node as having failed and inform GPFS to perform node recovery.

In all environments, each descriptor for a GPFS nodeset must be in the form:

```
NodeName[::manager|client]
```

Where `NodeName` is the hostname or IP address of the adapter to be used for GPFS communications. The optional designation specifies whether or not the node should be included in the pool of nodes from which the file system manager is chosen. The default is not to have the node included in the pool.

**node number.** GPFS references node numbers in an environment specific manner. In an RSCT peer domain environment, the node number is obtained from the peer node resource class. In a HACMP/ES cluster environment, the node number is obtained from the global ODM.

**nodeset.** A GPFS nodeset is a group of nodes that all run the same level of GPFS code and operate on the same file systems. You have the ability to define more than one GPFS nodeset in the same GPFS cluster.

Network Shared Disks (NSDs). The GPFS function that allows application programs executing at different nodes of a GPFS cluster to access a raw logical volume as if it were local at each of the nodes. In actuality, the logical volume is local at only one of the nodes (the server node).

**P**

**peer domain.** A set of nodes configured for high availability by the RSCT configuration manager. Such a domain has no distinguished or master node. All nodes are aware of all other nodes, and administrative commands can be issued from any node in the domain. All nodes also have a consistent view of the domain membership. Contrast with "management domain" on page 74.

**persistent reserve.** Persistent reserve is a capability of the ANSI SCSI-3 architecture for interfacing with storage devices. Specifically, persistent reserve provides control of access from multiple host systems which is useful in recovery situations. To access a storage device which is configured to use persistent reserve, a host must register using a unique key. In the event of a perceived failure, another host system may preempt that access using that unique key which will result in the storage device not honoring attempts to read or write data on the device until the pre-empted system has re-registered. Software conventions exist in GPFS which only allow a pre-empted system to re-register after the recovery situation has been addressed. Contrast with "disk leasing" on page 73.

**primary GPFS cluster data server.** In a GPFS cluster, this refers to the primary GPFS cluster data server node for the GPFS configuration data.

**PSSP cluster environment.** The operation of GPFS based on the PSSP and IBM Virtual Shared Disk program products.

**Q**

**quorum.** The minimum number of nodes that must be running in order for the GPFS daemon to start.

For all nodesets consisting of three or more nodes, the *multi-node quorum* algorithm applies defining quorum as one plus half of the number of nodes in the GPFS nodeset.
For a two node nodeset, the single-node quorum algorithm can be applied allowing the GPFS daemon to continue operation despite the loss of the peer node.

**quota.** The amount of disk space and number of inodes assigned as upper limits for a specified user or group of users.

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

**Redundant Array of Independent Disks (RAID).** A set of physical disks that act as a single physical volume and use parity checking to protect against disk failure.

**recovery.** The process of restoring access to file system data when a failure has occurred. This may involve reconstructing data or providing alternative routing through a different server.

**replication.** The practice of creating and maintaining multiple file copies to ensure availability in the event of hardware failure.

**RSCT peer domain.** See "peer domain" on page 75.

**SSA.** Serial Storage Architecture. An expanded storage adapter for multi-processor data sharing in UNIX-based computing, allowing disk connection in a high-speed loop.

**SCSI.** Small Computer Systems Interface. An adapter supporting attachment of various direct-access storage devices.

**secondary GPFS cluster data server.** In a GPFS cluster, this refers to the backup server node for the GPFS configuration data (see "GPFS cluster data" on page 74).

**session failure.** The loss of all resources of a data management session due to the failure of the GPFS daemon on the session node.

**session node.** The node on which a data management session was created.

**single-node quorum.** In a two node nodeset, use of the single-node quorum algorithm allows the GPFS daemon to continue operating in the event only one node is available. Use of this quorum algorithm is not valid if more than two nodes have been defined in the nodeset. This applies only in either an AIX cluster or Linux environment where disks are directly attached.

**source node.** The node on which a data management event is generated.

**stripe group.** The set of disks comprising the storage assigned to a file system.

**striping.** A method of writing a file system, in parallel, to multiple disks instead of to single disks in a serial operation.

**subblock.** The smallest unit of data accessible in an I/O operation, equal to one thirty-second of a data block.

**token management.** A system for controlling file access in which each application performing a read or write operation is granted exclusive access to a specific block of file data. This ensures data consistency and controls conflicts.

Token management has two components: the token manager server, located at the file system manager node, and the token management function on each node in the GPFS nodeset. The token management server controls tokens relating to the operation of the file system. The token management function on each node, including the file system manager node, requests tokens from the token management server.

**twin-tailing.** Connecting a disk to multiple nodes

**virtual file system (VFS).** A remote file system that has been mounted so that it is accessible to the local user. The virtual file system is an abstraction of a physical file system implementation. It provides a consistent interface to multiple file systems, both local and remote. This consistent interface allows the user to view the directory tree on the running system as a single entity even when the tree is made up of a number of diverse file system types.

**virtual shared disk.** See "IBM Virtual Shared Disk" on page 74.

**virtual node (vnode).** The structure which contains information about a file system object in a virtual file system.
Bibliography

This bibliography contains references for:

- GPFS publications
- AIX publications
- RSCT publications
- HACMP/ES publications
- IBM Subsystem Device Driver, IBM 2105 Enterprise Storage Server, and Fibre Channel
- IBM RedBooks
- Non-IBM publications that discuss parallel computing and other topics related to GPFS

All IBM publications are also available from the IBM Publications Center at [www.ibm.com/shop/publications/order](http://www.ibm.com/shop/publications/order)

GPFS publications

You may download, view, search, and print the supporting documentation for the GPFS program product in the following ways:

1. In PDF format:
   - From the IBM Publications Center at [www.ibm.com/shop/publications/order](http://www.ibm.com/shop/publications/order)

To view the GPFS PDF publications, you need access to the Adobe Acrobat Reader. The Acrobat Reader is shipped with the AIX 5L Bonus Pack and is also freely available for downloading from the Adobe web site at [www.adobe.com](http://www.adobe.com) Since the GPFS documentation contains cross-book links, if you choose to download the PDF files they should all be placed in the same directory and the files should not be renamed.

To view the GPFS HTML publications, you need access to an HTML document browser such as Netscape. An index file into the HTML files (aix_index.html) is provided when downloading the tar file of the GPFS HTML publications. Since the GPFS documentation contains cross-book links, all files contained in the tar file should remain in the same directory.

In order to use the GPFS man pages the gpfsdocs file set must first be installed (see Installing the GPFS man pages).

The GPFS library includes:

- General Parallel File System for AIX 5L: AIX Clusters Administration and Programming Reference, SA22-7896 (PDF file name an2adm10.pdf)
- General Parallel File System for AIX 5L: AIX Clusters Problem Determination Guide, GA22-7897 (PDF file name an2pdg10.pdf)
- General Parallel File System for AIX 5L: AIX Clusters Data Management API Guide, GA22-7898 (PDF file name an2dmp10.pdf)

AIX publications


© Copyright IBM Corp. 2002
Reliable Scalable Cluster Technology publications
You can download the RSCT related documentation from the web at
www.ibm.com/servers/eserver/clusters/library/:
- RSCT for AIX 5L: Guide and Reference, SA22-7889
- RSCT for AIX 5L: Messages, GA22-7891
- RSCT for AIX 5L: Technical Reference, SA22-7890
- RSCT Group Services Programming Guide and Reference, SA22-7888

HACMP/ES publications
You can download the HACMP/ES manuals from the Web at
www.ibm.com/servers/eserver/pseries/library/hacmp_docs.html
- HACMP for AIX 4.4 Enhanced Scalability Installation and Administration Guide, SC23-4306

Storage related information
Various references include:
- IBM TotalStorage at www.storage.ibm.com/sgg.html
- IBM Serial Storage Architecture support at www.storage.ibm.com/hardsoft/products/ssa/

Redbooks
IBM’s International Technical Support Organization (ITSO) has published a number of redbooks. For a current list, see the ITSO Web site at www.ibm.com/redbooks
- IBM @server Cluster 1600 and PSSP 3.4 Cluster Enhancements, SG24-6604 provides information on GPFS 1.5.
- GPFS on AIX Clusters; High Performance File System Administration Simplified, SG24-6035 provides information on GPFS 1.4.
- Implementing Fibre Channel Attachment on the ESS, SG24-6113
- Configuring and Implementing the IBM Fibre Channel RAID Storage Server, SG24-5414

Whitepapers
IBM @server pSeries white papers at www.ibm.com/servers/eserver/pseries/library/wp_systems.html
Clustering technology white papers at www.ibm.com/servers/eserver/pseries/library/wp_clustering.html
AIX white papers at www.ibm.com/servers/aix/library/wp_aix.html
Non-IBM publications

Here are some non-IBM publications that you may find helpful:

• Foster, I., *Designing and Building Parallel Programs*, Addison-Wesley, 1995.
Index

Special characters
/etc/security/limits 36
nofiles file descriptor limit 36

Numerics
64-bit support 37

A
access to the same file simultaneous 3
adapter membership 51, 52
administration commands
GPFS 51
AIX 6
communication with GPFS 51
AIX 5L 9
AIX cluster environment
description 6
allocation map
block 48
inode 48
logging of 49
application programs
communicating with GPFS 52
application support 59
autoload option 17
automatic mount
file systems 21
automount feature 4

B
bandwidth
increasing aggregate 3
block
allocation map 48
block size 19, 21
affect on maximum mounted file system size 22

C
cache 18, 50
cluster
restrictions 63
cluster type 15
clusters see GPFS cluster environment 83
coeexistence
conflicting software 29
coeexistence guidelines 39
commands
error communication 51
failure of 14
GPFS administration 51
mmadddisk 13
commands (continued)
mmchconfig 15, 37, 50
mmchdisk 56
mmcheckquota 24, 52
mmchfs 20, 37, 40, 48
mmconfig 15, 16, 33, 50
mmcrcluster 9, 14, 16, 29
mmcrfs 13, 20, 33
mmcrev 11
mmdefedquota 24
mmdefquotaon 24
mmedquota 23, 24
mmfsck 49, 52, 56
mmilsdisk 56
mmisquota 24
mmreppquota 24
mmrpldisk 13
mmstartup 17
operating system 52
processing 56
remote file copy
rcp 15
remote shell
rsh 15
restrictions 64
where they run 4
communication
between GPFS and RSCT 14
GPFS daemon to daemon 14
communication protocol 50
communications I/O 35
compatibility 40
configuration
file system manager nodes 45
files 57
of a GPFS cluster 14
options
all environments 15
system 68
system flexibility 4
configuration see also nodeset 83
configuration files 4
configuration manager 45, 52
configuration settings 35
configuring GPFS 15
conflicting software 29
considerations for GPFS applications 59
creating GPFS directory
/tmp/gpfslpp 31
cssMembership 52

D
daemon memory 49
data
availability 3
data (continued)
  consistency of 3
data blocks
  logging of 49
  recovery of 49
Data Management API (DMAPi)
  configuration options 15, 19
  enabling 24
data recoverability 9
default quotas 24
  files 49
definition
  of failure group 3
dercriptor
  file systems 47
descriptors
  disk 13
directives
  restrictions 67
disk descriptors 13
disk leasing 11
disk properties
  DiskUsage 11
  Failure Group 11
disk subsystems 9
disks
  descriptors 25
  failure 10, 11
  fencing 11
  media failure 57
  recovery 56
  releasing blocks 57
  restrictions 66
  state of 56
  tuning parameters 36
  usage 12, 25
  usage verification 24
DiskUsage
disk properties 11
documentation 31
  obtaining 77
dumps
  path for the storage of 17

E
electronic license agreement 29
estimated node count 21

F
failing nodes
  in multi-node quorum 18
  in single-node quorum 18
failover support 3
failure
  disk 10
  node 10, 18
failure group
  definition of 3

Failure Group
  disk properties 11
failure groups
  choosing 13, 25
file system manager 17
  administration command processing 51
  command processing 56
  communication with 51
  description 45
  mount of a file system 52
  pool of nodes to choose from 16
  selection of 46
file systems
  administrative state of 4, 57
  automatic mount of 21
  block size 19, 21
  creating 20
descriptor 47
device name 25
disk descriptor 25
interacting with a GPFS file system 52
maximum number of 48, 52
maximum number of files 22, 48
maximum size supported 48
mounted file system sizes 22
mounting 24, 52
opening a file 53
reading a file 53
recovery 57
repairing 56
restrictions 63
sizing 21
writing to a file 54
files
  /.rhosts 35
  /etc/cluster.nodes 58
  /etc/filesystems 57
  /etc/fstab 58
  /var/adm/ras/mmfs.log.latest 51
  /var/mmfs/etc/cluster.preferences 58
  /var/mmfs/etc/mmfs.cfg 58
  /var/mmfs/gen/mmsdrfs 58
  /etc/security/limits 36
  consistency of data 3
  group.quota 49
  inode 48
  log files 49
  maximum number of 22, 48
  maximum size 48
  mmfs.cfg 57
  structure within GPFS 47
  user.quota 49
  fragments, storage of files 22

G
GPFS
  administration commands 51
  communication within 51
daemon description 6
description of 3
GPFS (continued)
   nodeset in an HACMP environment  7
   nodeset in an RSCT peer domain environment  7
   planning for  9
   strengths of  3
   structure of  5, 45
GPFS cl data
   server nodes  14
GPFS cluster
   configuration restrictions  61
   creating  14
   defining nodes in the cluster  14
   planning nodes  14
GPFS cluster data
   content  4, 57
   designation of server nodes  14
GPFS daemon
   quorum requirement  45
   went down  51
   grace period, quotas  24
Group Services  10, 50
   initialization of GPFS  52
   recovering a file system  57

H
   ha.vsd group
      initialization of GPFS  52
   HACMP environment  7
   HACMP/ES
      HACMP environment  6
      HACMP/ES program product  9
   hard limit, quotas  24
   hardware specifications  9
   hints
      restrictions  67

I
   IBM Multi-Media Server
      conflicting software  29
   IBM Video Charger
      conflicting software  29
   indirect blocks  47, 49
   indirection level  47
   initialization of GPFS  52
   inode
      allocation file  48
      allocation map  48
      cache  50
      logging of  49
      usage  47, 55
   installation
      files used during  29
      images  32
      installing on a network  32
      on a non-shared file system network  33
      on a shared file system network  32
      verifying  33
      what to do after the installation of GPFS  33
      installation procedure  31

installing
   what to do before you install GPFS  29
   invariant IP address  35
   ipqmaxlen parameter  35

K
   kernel extensions  5
   kernel memory  49

L
   license inquiries  69
   load
      balancing across disks  3
   log files
      creation of  49
      unavailable  57
   logical volume
      creation considerations  11
   loose cluster
      cluster type  15

M
   man pages
      obtaining  77
   max_coalesce parameter  36
   maxFilesToCache
      memory usage for  19
   maxFilesToCache parameter  18, 50
   maximum number of files  22
   maxStatCache
      memory usage for  19
   maxStatCache parameter  18, 50
   memory
      controlling  18
      usage  49
   memory formula
      for maxFilesToCache  19
      for maxStatCache  19
   metadata  47
      disk usage to store  12, 25
   metanode  47
   migration
      full  38
      nodesets  37
      requirements  37
      reverting to the previous level of GPFS  39
      staged  37
   mmadddisk command
      and rewritten disk descriptor file  13
   mmcrfs command
      and rewritten disk descriptor file  13
   mmcrlv command  11
   mmrpldisk command
      and rewritten disk descriptor file  13
   mount command  52
   mounting a file system  24
   multi-node quorum  18
N
Network Shared Disks (NSDs)
definition 75
nodes
acting as special managers 45
estimating the number of 21
failure 10, 18, 57
in a GPFS cluster 14
planning 16
restrictions 64
nodeset
configuration restrictions 62
nodesets
creating 16
definition of 3
designation of 25
file for installation 29
identifier 17
in an HACMP environment 7
in an RSCT peer domain environment 7
migrating 37
moving a file system 25
operation of 17
planning 16
non-shared file system network
installing GPFS 33
notices 69

O
operating system
commands 52
operating system calls 53

P
pagepool
in support of I/O 50
pagepool parameter
affect on performance 54
usage 18, 50
parameter
maxStatCache 18
parameters
maxFilesToCache 18
patent information 69
PATH environment variable 29
performance
pagepool parameter 54
use of GPFS to improve 3
use of pagepool 50
performance improvements
balancing load across disks 3
increasing aggregate bandwidth 3
parallel processing 3
simultaneous access the same file 3
supporting large amounts of data 3
persistent reserve 11
pool of nodes
in selection of file system manager 46
programming interfaces, use of 64-bit 37
programming specifications 9
conflicting software 29
verifying 30
properties
disk
DiskUsage 11
Failure Group 11
PVID
verification of 11

Q
quorum
definition of 18
during node failure 10
enforcement 45
initialization of GPFS 52
quotas
default quotas 24
description 23
files 49
in a replicated system 23
mounting a file system with quotas enabled 24
role of file system manager node 46
system files 24
values reported in a replicated file system 23

R
rcp 15
read operation
buffer available 53
buffer not available 54
requirements 53
token management 54
README file, viewing 32
recoverability 11
disk failure 10
disks 56
features of GPFS 3, 57
file systems 56
node failure 10
recoverability parameters 9
Redundant Array of Independent Disks (RAID) 12
Reliable Scalable Cluster Technology (RSCT)
subsystem of AIX 6
remote file copy command
rcp 15
remote shell command
rsh 15
removing GPFS 41
repairing a file system 56
replication 11
affect on quotas 23
description of 4
restrictions
cluster management 63
commands 64
disk management 66
file system configuration 63
restrictions (continued)

GPFS cluster configuration 61
node management 64
nodeset configuration 62
starting GPFS 62
restripe see rebalance 83
rewritten disk descriptor file uses of 13
RSCT peer domain environment 7
rsh 15

S

SCSI-3 persistent reserve 11
secondary network for RSCT communications 14
security 35
  GPFS use of 46
  restrictions 64
shared external disks
  considerations 9
shared file system network
  installing GPFS 32
shared segments 50
single-node quorum 18
sizing file systems 21
socket communications, use of 51
soft limit, quotas 24
softcopy documentation 31
SSA fencing 11
SSA Redundant Array of Independent Disks (RAID) 22
standards, exceptions to 59
starting GPFS 17
  restrictions 62
stat cache 50
stat() system call 50, 55
storage see memory 83
Stripe Group Manager see File System Manager 83
structure of GPFS 5
subblocks, use of 22
support
  failover 3
syntax
  rcp 62
  rsh 62
system calls
  open 53
  read 53
  stat() 55
  write 54
system configuration 68
System Data Repository (SDR)
  configuring all of nodes listed in 16

T

token management
  description 46
  system calls 53
token management system 3
topology services
  configuration settings 35

trademarks 70
Transmission Control Protocol/Internet Protocol (TCP/IP) 50
tuning parameters
  ipqmaxlen 35
  max_coalesce 36
tuning your system 35
two-node nodeset 18

U

uninstalling GPFS 41
user data 49

V

verification
  disk usage 24
  verifying prerequisite software 30

W

write operation
  buffer available 55
  buffer not available 55
token management 55
Readers’ Comments — We’d Like to Hear from You

General Parallel File System for AIX 5L
AIX Clusters Concepts, Planning, and Installation Guide

Publication No. GA22-7895-01

Overall, how satisfied are you with the information in this book?

<table>
<thead>
<tr>
<th>Overall satisfaction</th>
<th>Very Satisfied</th>
<th>Satisfied</th>
<th>Neutral</th>
<th>Dissatisfied</th>
<th>Very Dissatisfied</th>
</tr>
</thead>
</table>

How satisfied are you that the information in this book is:

<table>
<thead>
<tr>
<th>Information Type</th>
<th>Very Satisfied</th>
<th>Satisfied</th>
<th>Neutral</th>
<th>Dissatisfied</th>
<th>Very Dissatisfied</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accurate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to find</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy to understand</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Well organized</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Applicable to your tasks</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Please tell us how we can improve this book:

Thank you for your responses. May we contact you?  □ Yes  □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name

Address

Company or Organization

Phone No.