General Parallel File System

Concepts, Planning, and Installation Guide

Version 3 Release 4

GA76-0413-04

General Parallel File System

Concepts, Planning, and Installation Guide

Version 3 Release 4

GA76-0413-04

Note
Before using this information and the product it supports, read the information in "Notices" on page 105.

This edition applies to version 3 release 4 of IBM General Parallel File System Multiplatform (program number 5724-N94), IBM General Parallel File System for POWER® (program number 5765-G66), and to all subsequent releases and modifications until otherwise indicated in new editions. Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

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

Contents

Figures

Tables

About this information

The *General Parallel File System: Concepts, Planning, and Installation Guide* provides information about these topics:

- Introducing GPFS™
- Planning concepts for GPFS
- SNMP support
- Installing GPFS
- Migration, coexistence and compatibility
- Applying maintenance
- Configuration and tuning
- Steps to uninstall GPFS

This edition applies to GPFS version 3.4 for AIX®, Linux®, and Windows®.

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

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

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

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

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

Who should read this information

This information is intended for system administrators, analysts, installers, planners, and programmers of GPFS clusters.

It assumes that you are very experienced with and fully understand the operating systems on which your cluster is based.

Use this information if you are:

- Planning for GPFS
- Installing GPFS on a supported cluster configuration, consisting of:
  - Linux nodes
  - AIX nodes
  - Windows nodes
  - An interoperable cluster comprised of all operating systems

Conventions used in this information

Table 1 on page x describes the typographic conventions used in this information.

GPFS for Windows note: UNIX® file name conventions are used throughout this information. For example, the GPFS cluster configuration data is stored in the **/var/mmfs/gen/mmsdrfs** file. On Windows, the UNIX name space starts under the **%SystemRoot%\SUA** directory, and UNIX-style file names need to be converted accordingly. For example, the cluster configuration file mentioned above is **C:\Windows\SUA\var\mmfs\gen\mmsdrfs**.

Table 1. Conventions

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

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

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

Prerequisite and related information

For updates to this information, see the GPFS library at (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfsbooks.html).

For the latest support information, see the GPFS Frequently Asked Questions at (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

How to send your comments

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

| **mhvrcfs@us.ibm.com**

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

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

| **gpfs@us.ibm.com**

Summary of changes

This topic summarizes changes to the GPFS licensed program and the GPFS library for version 3, release 4. Within each information unit in the library, a vertical line to the left of text and illustrations indicates technical changes or additions made to the previous edition of the book.

Important: Nodes running GPFS 3.4 can coexist and interoperate with nodes running GPFS 3.3. GPFS 3.4 is not compatible with GPFS 3.2 or earlier versions. After you install GPFS 3.4 on some of the nodes in the cluster, nodes that are still running GPFS 3.2 or earlier will not be able to join the cluster. Similarly, in a multicluster environment in which clusters are remotely mounting file systems from other clusters, if one of the clusters is migrated to GPFS 3.4, the remaining clusters are expected to be at GPFS 3.3 or later.

Changes to the GPFS licensed program for version 3, release 4 include the following:

- Enhanced Windows cluster support:
  - Windows Server 2008 R2 x64
  - Directly attached disks support higher bandwidth via local I/O

    The use of fully SAN-connected Windows clusters may enable much higher bandwidth to Windows systems using GPFS, and SAN connectivity may provide greater flexibility in configuring GPFS clusters.
  - Homogeneous Windows clusters

    GPFS clusters can now be formed using only Windows nodes; Linux or AIX are no longer required as NSD servers. The Windows nodes can perform most of the required management and administrative operations. The exceptions include:
    - Certain GPFS commands to apply policy, administer quotas and ACLs.
    - The ability to mount DMAPI-enabled file systems.
    - Support for Tivoli® Storage Manager (TSM) Backup Archive client or the native Windows Backup utility.
- Performance and scaling improvements:

  Extended attributes for a file can now be stored in the file's inode. This can improve the performance of applications that use extended attributes, and reduce the amount of disk space required to store them. This change enables the high-performance GPFS policy engine to refer to extended attributes with the same scale-out performance as the regular file attributes, providing richer semantics to content managers and middleware.
- Migration improvements include the new **mmmigratefs** command, which you can use for migrating file systems.
- Diagnostic improvements include the new **mmdiag** command, which better enables IBM Service to determine the state of your GPFS cluster.
- Support for more than 2,147,483,648 files in a file system. For the current maximum tested limit, see the GPFS FAQ at:

  http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfs_faqs.html
- Withdrawal of support for the following:
  - data shipping mode
  - 32-bit AIX kernels
  - the High Performance Switch
  - IBM Virtual Shared Disks

    New file systems must be created using network shared disks (NSDs) only.

- The GPFS Graphical User Interface (GUI)
- New commands:
  - **mmdiag**
  - **mmmigratefs**
- New subroutines:
  - **dm_remove_dmattr_nosync**
  - **dm_set_dmattr_nosync**
  - **dm_set_nolist_nosync**
  - **dm_set_region_nosync**
  - **dm_sync_dmattr_by_handle**
  - **gpfs_igetattrsx**
  - **gpfs_iputattrsx**
- New structures:
  - There are no new structures.
- Changed commands:
  - **mmaddcallback**
  - **mmadddisk**
  - **mmapplypolicy**
  - **mmchattr**
  - **mmchconfig**
  - **mmchdisk**
  - **mmcheckquota**
  - **mmchfs**
  - **mmcrfs**
  - **mmdefragfs**
  - **mmdeldisk**
  - **mmdelsnapshot**
  - **mmfsck**
  - **mmfsctl**
  - **mmlscallback**
  - **mmlsfs**
  - **mmlssnapshot**
  - **mmremotefs**
  - **mmrestoreconfig**
  - **mmrestripefs**
  - **mmrpldisk**
- Changed DMAPI configuration options:
  - **dmapiFileHandleSize**
- Changed error log entries:
  - **MMFS_LONGDISKIO**
  - **MMFS_SYSTEM_UNMOUNT**
  - **MMFS_SYSTEM_WARNING**
- Changed structures:
  - There are no changed structures.
- Messages:

Table 2. New, changed, and deleted messages

New messages	Changed messages	Deleted messages
6027-1515	6027-302, 6027-303, 6027-304, 6027-305, 6027-340, 6027-470, 6027-475, 6027-578, 6027-591, 6027-1133, 6027-1500, 6027-1507, 6027-1508, 6027-1509, 6027-1544, 6027-2024, 6027-2028, 6027-2029, 6027-2030, 6027-2031, 6027-2166, 6027-2117, 6027-2150, 6027-2151, 6027-2152, 6027-2153, 6027-2154, 6027-2155, 6027-2156, 6027-2157, 6027-2158, 6027-2601, 6027-2608, 6027-2667, 6027-2682, 6027-2732, 6027-2733, 6027-2734, 6027-2735, 6027-2736, 6027-2951, 6027-3005, 6027-3006, 6027-3007, 6027-3100	6027-533, 6027-2502, 6027-2679

The "Readers' Comments - We'd Like to Hear from You" section at the back of this publication has been replaced with the topic “How to send your comments” on page x.

Chapter 1. Introducing General Parallel File System

The IBM General Parallel File System (GPFS) is a cluster file system. This means that it provides concurrent access to a single file system or set of file systems from multiple nodes. These nodes can all be SAN attached or a mix of SAN and network attached. This enables high performance access to this common set of data to support a scale-out solution or provide a high availability platform.

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

The strengths of GPFS

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

For more information, see the following:

Shared file system access among GPFS clusters

GPFS allows you to share data between separate clusters within location or across a WAN. Between clusters, users can share access to some or all file systems in either cluster. Access between clusters can be used for administrative purposes; for example a cluster may not own any storage, but may mount file systems only from another cluster. This configuration allows for separate management of GPFS clusters. This configuration is referred to as a multi-cluster environment using the GPFS remote mount feature. When multiple clusters are configured to access the same GPFS file system, Open Secure Sockets Layer (OpenSSL) is used to authenticate and check authorization for all network connections.

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

GPFS multi-cluster environment provides for:

- The ability of the cluster hosting the file system to specify different security levels, for each cluster authorized to mount a particular file system.
- A highly available service as the local cluster may remain active prior to changing security keys. Periodic changing of keys is necessary for a variety of reasons, including:

– In order to make connection rate performance acceptable in large clusters, the size of the security keys used for authentication cannot be very large. As a result it may be necessary to change security keys in order to prevent a given key from being compromised while it is still in use.
– As a matter of policy, some institutions may require security keys are changed periodically.

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

Improved system performance

Using GPFS file systems can improve system performance by:

- Allowing multiple processes or applications on all nodes in the cluster simultaneous access to the same file using standard file system calls.
- Increasing aggregate bandwidth of your file system by spreading reads and writes across multiple disks.
- Balancing the load evenly across all disks to maximize their combined throughput, eliminating storage hotspots.
- Supporting very large file and file system sizes.
- Allowing concurrent reads and writes from multiple nodes.
- Allowing for distributed token (lock) management. Distributing token management reduces system delays associated with a lockable object waiting to obtaining a token. Refer to "Increased data availability" on page 3 for additional information on token management.
- Allowing for the specification of multiple networks for GPFS daemon communication and for GPFS administration command usage within your cluster.

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

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

GPFS I/O performance may be monitored through the **mmpmon** command. See the *GPFS: Advanced Administration Guide*.

File consistency

GPFS uses a sophisticated token management system to provide data consistency while allowing multiple independent paths to the same file by the same name from anywhere in the cluster. See Chapter 2, "GPFS architecture," on page 9.

Increased data availability

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

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

During configuration, you assign a replication factor to indicate the total number of copies of data and metadata you wish to store. Currently the maximum replication factor is 2, indicating that two copies of data and metadata should be kept for replicated files. Replication allows you to set different levels of protection for each file or one level for an entire file system. Since replication uses additional disk space and requires extra write time, you should consider the impact of replication on your application, especially if the replication is occurring over a WAN. To reduce the overhead involved with the replication of data, you may also choose to replicate only metadata as a means of providing additional file system protection. For further information on GPFS replication, see "File system replication parameters" on page 51.

GPFS is a logging file system and creates separate logs for each node. These logs record the allocation and modification of metadata aiding in fast recovery and the restoration of data consistency in the event of node failure. Even if you do not specify replication when creating a file system, GPFS automatically replicates recovery logs in separate failure groups, if multiple failure groups are available. This replication feature can be used in conjunction with other GPFS capabilities to maintain one replica in a geographically separate location which provides the capability to survive disasters at the other location. For further information on failure groups, see "Network Shared Disk (NSD) creation considerations" on page 40. For further information on disaster recovery with GPFS see the *GPFS: Advanced Administration Guide*.

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

Enhanced system flexibility

With GPFS, your system resources are not frozen. You can add or delete disks while the file system is mounted. When the time is right and system demand is low, you can rebalance the file system across all currently configured disks. In addition, you can also add or delete nodes without having to stop and restart the GPFS daemon on all nodes.

Note: In the node quorum with tiebreaker disk configuration, GPFS has a limit of eight quorum nodes. If you add quorum nodes and exceed that limit, the GPFS daemon must be shutdown. Before you restart the daemon, switch quorum semantics to node quorum. For additional information, refer to "Quorum" on page 29.

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

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

Simplified storage management

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

- Storage pools
- Policies
- Filesets

Storage pools

A *storage pool* is a collection of disks or RAIDs with similar properties that are managed together as a group. Storage pools provide a method to partition storage within a file system. While you plan how to configure your storage, consider factors such as:

- Improved price-performance by matching the cost of storage to the value of the data.
- Improved performance by:
  - Reducing the contention for premium storage
  - Reducing the impact of slower devices
- Improved reliability by providing for:
  - Replication based on need
  - Better failure containment

Policies

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

Placement policies:

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

File management policies

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

Filesets

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

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

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

Simplified administration

GPFS offers many of the standard file system interfaces allowing most applications to execute without modification. Operating system utilities are also supported by GPFS. That is, you can continue to use the commands you have always used for ordinary file operations (see Chapter 11, "Considerations for GPFS applications," on page 101). The only unique commands are those for administering the GPFS file system.

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

GPFS commands save configuration and file system information in one or more files, collectively known as GPFS cluster configuration data files. The GPFS administration commands keep these files synchronized between each other and with the GPFS system files on each node in the cluster, thereby providing for accurate configuration data. See "Cluster configuration data files" on page 24.

The basic GPFS structure

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

For more information, see the following topics:

1. "GPFS administration commands"
2. "The GPFS kernel extension"
3. "The GPFS daemon" on page 6
4. For nodes in your cluster operating with the Linux operating system, "The GPFS open source portability layer" on page 7

For a detailed discussion of GPFS, see Chapter 2, "GPFS architecture," on page 9.

GPFS administration commands

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

By default, GPFS commands can be executed from any node in the cluster. If tasks need to be done on another node in the cluster, the command automatically redirects the request to the appropriate node for execution. For administration commands to be able to operate, passwordless remote shell communications between the nodes is required. This can be accomplished using a standard shell program like **rsh** or **ssh** or a custom application that uses the same semantics. To provide additional security, you can control the scope of passwordless access by allowing administration tasks to be performed either from all nodes running GPFS (**mmchconfig adminMode=allToall**) or from a subset of nodes (**mmchconfig adminMode=central**). The default mode for clusters created with GPFS 3.3 or later is **adminMode=central**.

For more information, see the **mmchconfig** command in the *GPFS: Administration and Programming Reference*.

The GPFS kernel extension

The GPFS kernel extension provides the interfaces to the operating system vnode and virtual file system (VFS) layer to register GPFS as a native file system. Structurally, applications make file system calls to the operating system, which presents them to the GPFS file system kernel extension. GPFS uses the standard mechanism for the operating system. In this way, GPFS appears to applications as just another file

system. The GPFS kernel extension will either satisfy these requests using resources which are already available in the system, or send a message to the GPFS daemon to complete the request.

The GPFS daemon

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

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

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

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

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

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

Note: GPFS assumes that nodes within a subnet are connected using high-speed networks. For additional information on subnet configuration, refer to "Using public and private IP addresses for GPFS nodes" on page 16.

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

Note: To manually cause this discovery action, use the **mmnsddiscover** command. For more information about the **mmnsddiscover** command, see the *GPFS: Administration and Programming Reference*.

The default order of access used during NSD discovery follows:

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

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

It is suggested that you always define NSD servers for the disks. In a SAN configuration where NSD servers have also been defined, if the physical connection is broken, GPFS dynamically switches to the

server nodes and continues to provide data. GPFS falls back to local disk access when it has discovered the path has been repaired. This is the default behavior, and it can be changed with the useNSDserver file system mount option.

For further information, see "Disk considerations" on page 39 and "NSD disk discovery" on page 23.

The GPFS open source portability layer

On Linux platforms, GPFS uses a loadable kernel module that enables the GPFS daemon to interact with the Linux kernel. Source code is provided for the portability layer so that the GPFS portability can be built and installed on a wide variety of Linux kernel versions and configuration. When installing GPFS on Linux, you build a portability module based on your particular hardware platform and Linux distribution to enable communication between the Linux kernel and GPFS. See "Building the GPFS portability layer" on page 64.

GPFS cluster configurations

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

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

- Application I/O performance and reliability requirements
- The properties of the underlying storage hardware.
- Administration, Security and ownership considerations

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

There are four basic GPFS configurations:

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

All nodes attached to a common set of LUNS

In this type of configuration, all of the nodes in the GPFS cluster are connected to a common set of LUNS (for example, over a SAN). The following are some areas to consider with this configuration:

- The maximum number of nodes accessing a LUN you want to support
- Whether the storage supports mixed OS access to common LUN
- The fact that you cannot mix Windows nodes SAN access with Linux or AIX

For an example, see Figure 1 on page 8.

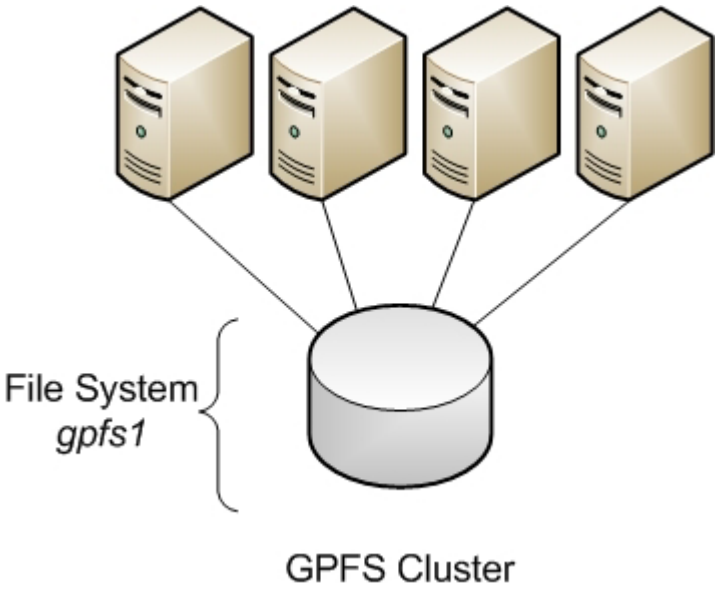

Figure 1. A cluster with disks that are SAN-attached to all nodes

GPFS Servers and GPFS clients

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

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

Sharing data across multiple GPFS clusters

GPFS allows you to share data across multiple GPFS clusters. Once a file system is mounted in another GPFS cluster all access to the data is the same as if you were in the host cluster. You can connect multiple clusters within the same data center or across long distances over a WAN. In a multicluster configuration, each cluster can be placed in a separate administrative group simplifying administration or provide a common view of data across multiple organizations.

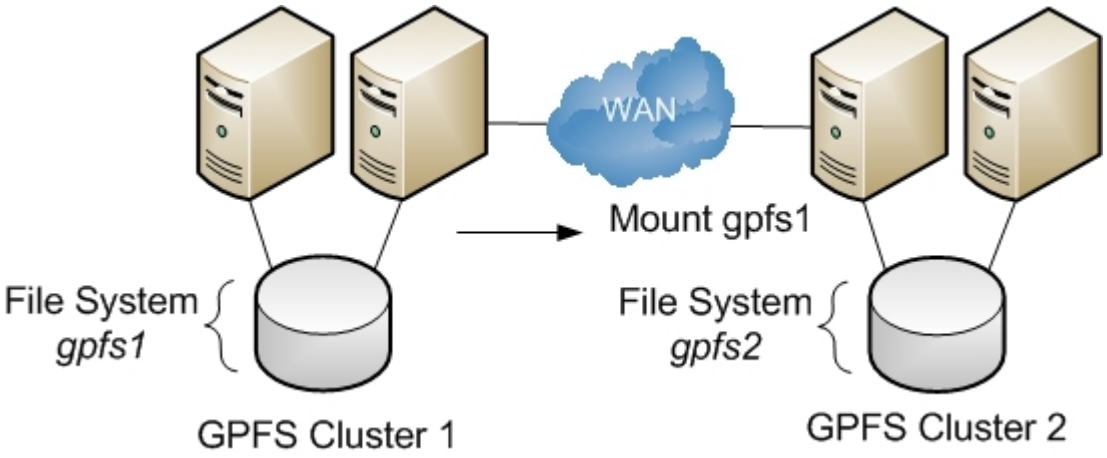

Figure 2. A multicluster configuration

Chapter 2. 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

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:

The GPFS cluster manager

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

Note: See "Quorum" on page 29 for more information.
The cluster manager performs the following tasks:

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

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

  The cluster manager prevents multiple nodes from assuming the role of file system manager, thereby avoiding data corruption, as the token management function resides on the file system manager node and possibly other nodes. See *Using multiple token servers* in the *General Parallel File System: Advanced Administration Guide*.
- Handles UID mapping requests from remote cluster nodes.

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

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

The file system manager

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

1. File system configuration

   Processes 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 file system manager node may also perform the duties of the token manager server. If you have explicitly designated some of the nodes in your cluster as file system manager nodes, then the token server load will be distributed among all of the designated manager nodes. For additional information, refer to *Using multiple token servers* in the *General Parallel File System: Advanced Administration Guide*.

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

   a. On the token management server

   b. On the token management client holding the token

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

   The normal flow for a token is:

   - A message to the token management server.

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

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

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

4. Quota management

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

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

   Notes:

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

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

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

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

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

The metanode

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

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

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

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

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

Each file has an inode containing information such as file size 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.

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

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

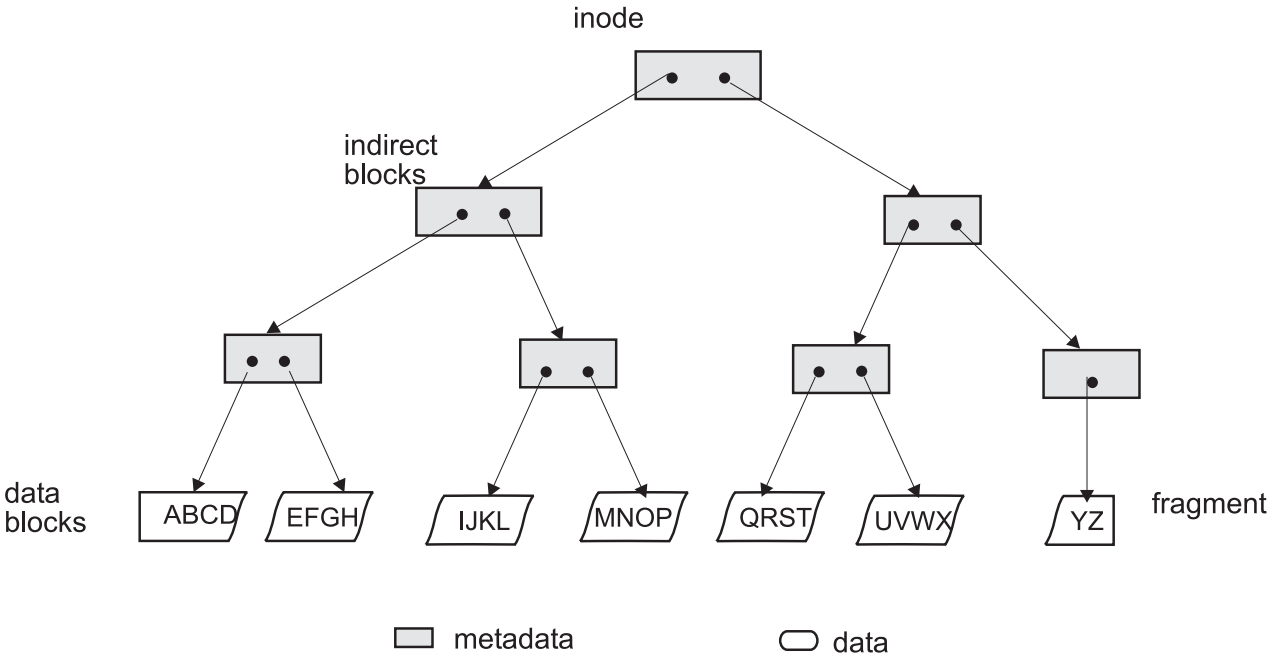

Figure 3. GPFS files have a typical UNIX structure

File system limitations:

1. The maximum number of mounted file systems within a GPFS cluster is 256.
2. See the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for the latest supported file system size.
3. The maximum number of files within a file system cannot exceed the architectural limit (for the current limit, see the GPFS FAQ at the URL listed above).

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

- Root directory of the file system
- *Block allocation map*, which is is a collection of bits that represent the availability of disk space within the disks of the file system. One unit in the allocation map represents a subblock or 1/32 of the block size of the file system. The allocation map is broken into regions that reside on disk sector boundaries.

The number of regions is set at file system creation time by the parameter that specifies how many nodes will access this file system. The regions are separately locked and, as a result, different nodes can be allocating or de-allocating space represented by different regions independently and concurrently.

- *Inode allocation map*, which represents the availability of inodes within the inode file. The *Inode allocation map* is located in the *inode allocation file*, and represents all the files, directories, and links that can be created. The **mmchfs** command can be used to change the maximum number of files that can be created in the file system up to the architectural limit.

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

Quota files

For file systems with quotas enabled, quota files are created at file system creation time. There are three quota files for a file system:

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

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

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

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

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

GPFS recovery logs

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

Recovery logs are replicated only if default metadata replication is turned on (**-m 2**). You can check to see if default replication is enabled for a file system using the **mmlsfs** command and looking at the value of the **-m** parameter or by setting **-m** to a value of 2 when creating or modifying a file system. When metadata is replicated, each node is given two recovery log files (Log Group); otherwise each node only has a single recovery log file.

When the metadata is replicated and a disk containing one of the recovery logs is stopped by a hardware error or by the **mmchdisk** command, the node stops using that copy of the log and just relies on the surviving copy. When the **mmrestripefs** command (with the **-m**, **-r**, or **-b** flag), the **mmchdisk start** command, or the **mmdeldisk** command is run, a new recovery log file is assigned by the file system manager to replace the log file on the disk that is down.

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

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

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

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

GPFS and memory

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

Memory allocated from the kernel heap

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

Memory allocated within the daemon segment

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

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

The file system manager is a token manager, and other nodes may assume token management responsibilities; therefore, any manager node may consume additional memory for token management. Refer to *Using multiple token servers* in the *General Parallel File System: Advanced Administration Guide*.

Shared segments accessed from both the daemon and the kernel

Shared segments consist of both pinned and unpinned memory that is allocated at daemon startup. The initial values are the system defaults. However, you can change these values later using the **mmchconfig** command. See "Cluster configuration file" on page 38.

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

1. A full inode cache
2. A stat cache

Pinned and non-pinned memory

Pinned memory

GPFS uses pinned memory (also called **pagepool** memory) for storing file data and metadata in support of I/O operations. With some access patterns, increasing the amount of **pagepool** memory can increase I/O performance. Increased pagepool memory can be useful in the following cases:

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

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

Non-pinned memory

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

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

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

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

1. Recently accessed files
2. Directories recently accessed by a number of **stat( )** calls

Note:

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

GPFS and network communication

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

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

GPFS daemon communication

In a cluster environment, the GPFS daemon depends on the correct operation of TCP/IP. These dependencies exist because:

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

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

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

Using public and private IP addresses for GPFS nodes

GPFS permits the system administrator to set up a cluster such that both public and private IP addresses are in use. For example, if a cluster has an internal network connecting some of its nodes, it is advantageous to use private IP addresses to communicate on this network, and public IP addresses to communicate to resources outside of this network. Public IP addresses are those that can be used to access the node from any other location for which a connection exists. Private IP addresses may be used only for communications between nodes directly connected to each other with a communications adapter. Private IP addresses are assigned to each node at hardware setup time, and must be in a specific address range (IP addresses on a 10.0.0.0, 172.16.0.0, or 192.168.0.0 subnet). For more information on private IP addresses refer to RFC 1597 - Address Allocation for Private Internets at http://www.ip-doc.com/rfc/rfc1597.

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

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

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

when the cluster was created or when the node was added to the cluster. For example, to add remote access to an existing cluster that was using only switch addresses, it is sufficient to add two **subnets** configuration parameters.

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

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

1. For each of its own IP addresses, it searches the other node's list of IP addresses for an address that is on the same subnet.
   - For normal public IP addresses this is done by comparing IP address values ANDed with the node's subnet mask for its IP address.
   - For private IP addresses GPFS assumes that two IP addresses are on the same subnet only if the two nodes are within the same cluster, or if the other node is in one of the clusters explicitly listed in the **subnets** configuration parameter.
2. If the two nodes have more than one IP address pair on a common subnet, GPFS uses the first one found according to the order of **subnets** specified in the initiating node's configuration parameters.
3. If there are no two IP addresses on the same subnet, GPFS uses the last IP address in each node's IP address list. In other words, the last subnet specified in the **subnets** configuration parameter is assumed to be on a network that is accessible from the outside.

For more information and an example, see *Using remote access with public and private IP addresses* in *General Parallel File System: Advanced Administration Guide*.

GPFS administration commands

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

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

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

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

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

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

Application and user interaction with GPFS

There are four ways to interact with a GPFS file system, which are outlined in this topic.

You can interact with a GPFS file system using:

- Operating system commands, which are run at GPFS daemon initialization time or at file system mount time (see "Operating system commands")
- Operating system calls such as **open()**, from an application requiring access to a file controlled by GPFS (see "Operating system calls")
- GPFS commands described in the *Commands* section of the *GPFS: Administration and Programming Reference* (see also"GPFS command processing" on page 22)
- GPFS programming interfaces described in the *Programming interfaces* section of the *GPFS: Administration and Programming Reference* and the *GPFS: Data Management API Guide*

Operating system commands

Operating system commands operate on GPFS data during:

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

Initialization of the GPFS daemon

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

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

The mounting of a file system

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

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

Operating system calls

The most common interface to files residing in a GPFS file system is through normal file system calls to the operating system. When a file is accessed, the operating system submits the request to the GPFS

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

Operating system calls operate on GPFS data during:

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

Opening a GPFS file

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

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

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

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

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

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

Reading file data

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

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

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

Buffer and locks available in memory:

The simplest **read** operation occurs when the data is already available in memory, either because it has been pre-fetched or because it has been read recently by another **read** call. In either case, the buffer is locally locked and the data is copied to the application data area. The lock is released when the copy is

complete. Note that no token communication is required because possession of the buffer implies that the node at least has a read token that includes the buffer. After the copying, prefetching is initiated if appropriate.

Tokens available locally but data must be read:

The second, more complex, type of **read** operation is necessary when the data is not in memory. This occurs under three conditions:

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

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

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

Data and tokens must be acquired:

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

Writing file data

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

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

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

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

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

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

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

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

Buffer available in memory:

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

Token available locally but data must be read:

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

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

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

Data and tokens must be acquired:

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

The stat system call

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

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

Inodes are cached within GPFS in two forms:

1. Full inode

2. Limited stat cache form

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

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

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

GPFS command processing

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

The mmchdisk command

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

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

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

The mmfsck Command

The **mmfsck** command repairs file system structures. **mmfsck** operates in two modes:

1. online
2. offline

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

The offline version of **mmfsck** is the last line of defense for a file system that cannot be used. It will most often be needed in the case where GPFS recovery logs are not available because of disk media failures. **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.

NSD disk discovery

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

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

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

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

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

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

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

Failure recovery processing

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

It should be noted that only one state change, such as the loss or initialization of a node, can be processed at a time and subsequent changes are queued. This means that the entire failure processing must complete before the failed node can join the group again. All failures are processed first, which means that GPFS handles all failures prior to completing any recovery.

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

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

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

Cluster configuration data files

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

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

The GPFS cluster configuration data is stored in the **/var/mmfs/gen/mmsdrfs** file. This file is stored on the nodes designated as the *primary GPFS cluster configuration server* and, if specified, the secondary GPFS cluster configuration server. See "GPFS cluster configuration servers" on page 36. The first record in the **mmsdrfs** file contains a generation number. Whenever a GPFS command causes something to change in the cluster or any of the file systems, this change is reflected in the **mmsdrfs** file and the generation number is increased by one. The latest generation number is always recorded in the **mmsdrfs** file on the primary and secondary GPFS cluster configuration server nodes.

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

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

Linux **/etc/fstab**

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

AIX **/etc/filesystems**

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

All GPFS nodes

/var/mmfs/gen/mmfsNodeData

Contains GPFS cluster configuration data pertaining to the node.

/var/mmfs/gen/mmsdrfs

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

/var/mmfs/gen/mmfs.cfg

Contains GPFS daemon startup parameters.

GPFS backup data

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

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

Chapter 3. Planning for GPFS

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

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

Planning for GPFS includes:

Hardware requirements

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

1. Consult the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for latest list of:
   - Supported server hardware
   - Tested disk configurations
   - Maximum cluster size
2. Provide enough disks to contain the file system. Disks can be:
   - SAN-attached to each node in the cluster
   - Attached to one or more NSD servers
   - A mixture of directly-attached disks and disks that are attached to NSD servers

   Refer to "Network Shared Disk (NSD) creation considerations" on page 40 for additional information.
3. When doing network-based NSD I/O, GPFS passes a large amount of data between its daemons. For NSD server-to-client traffic, it is suggested that you configure a dedicated high-speed network solely for GPFS communications when the following are true:
   - There are NSD disks configured with servers providing remote disk capability.
   - Multiple GPFS clusters are sharing data using NSD network I/O.

   Refer to the *GPFS: Advanced Administration Guide* for additional information.

GPFS communications require static IP addresses for each GPFS node. IP address takeover operations that transfer the address to another computer are not allowed for the GPFS network. Other IP addresses within the same computer that are not used by GPFS can participate in IP takeover. To provide availability or additional performance, GPFS can use virtual IP addresses created by aggregating several network adapters using techniques such as EtherChannel or channel bonding.

Software requirements

GPFS planning includes understanding the latest software requirements.

- GPFS is supported on AIX, Linux, and Windows.
- OpenSSL is required for remote cluster access.

Consult the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for the latest list of:

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

Recoverability considerations

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

- At the disk level, consider preparing disks for use with your file system by specifying failure groups that are associated with each disk. With this configuration, information is not vulnerable to a single point of failure. See "Network Shared Disk (NSD) creation considerations" on page 40.
- At the file system level, consider replication through the metadata and data replication parameters. See "File system replication parameters" on page 51.

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

1. "Node failure"
2. "Network Shared Disk server and disk failure" on page 32
3. "Reduced recovery time using Persistent Reserve" on page 33

Node failure

In the event of a node failure, GPFS:

- Prevents the continuation of I/O from the failing node
- Replays the file system metadata log for the failed node

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

To reduce the amount of time it takes for disk leases to expire, you can use Persistent Reserve (SCSI-3 protocol). If Persistent Reserve (configuration parameter: **usePersistentReserve**) is enabled, GPFS prevents the continuation of I/O from a failing node by fencing the failed node using a feature of the disk subsystem called Persistent Reserve. Persistent Reserve allows the failing node to recover faster because GPFS does not need to wait for the disk lease on the failing node to expire. For additional information, refer to "Reduced recovery time using Persistent Reserve" on page 33. For further information about recovery from node failure, see the *GPFS: Problem Determination Guide*.

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

Quorum

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

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

During node failure situations, quorum needs to be maintained in order for the cluster to remain online. If quorum is not maintained due to node failure, GPFS unmounts local file systems on the remaining nodes and attempts to reestablish quorum, at which point file system recovery occurs. For this reason it is important that the set of quorum nodes be carefully considered (refer to "Selecting quorum nodes" on page 31 for additional information).

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

- Node quorum
- Node quorum with tiebreaker disks

Node quorum: Node quorum is the default quorum algorithm for GPFS. With node quorum:

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

For example, in Figure 4 on page 30, there are three quorum nodes. In this configuration, GPFS remains active as long as there are two quorum nodes available.

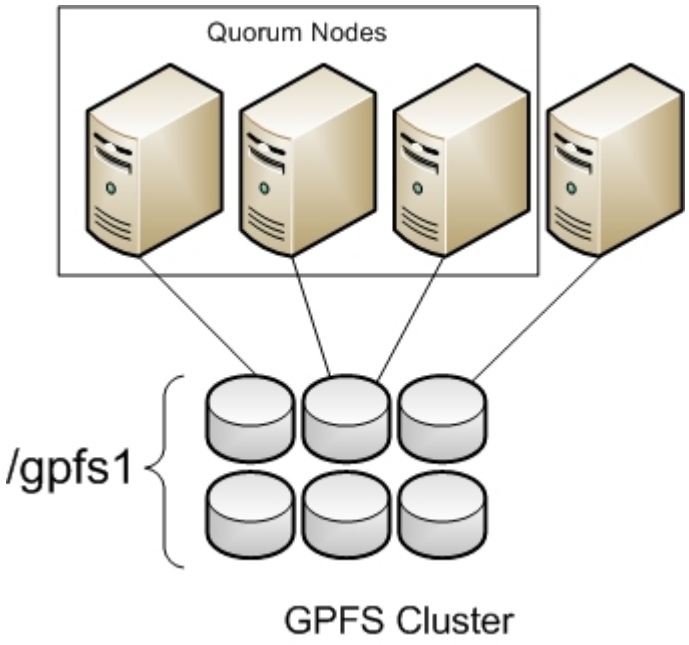

Figure 4. GPFS configuration using node quorum

Node quorum with tiebreaker disks: When running on small GPFS clusters, you might want to have the cluster remain online with only one surviving node. To achieve this, you need to add a tiebreaker disk to the quorum configuration. Node quorum with tiebreaker disks allows you to run with as little as one quorum node available as long as you have access to a majority of the quorum disks (refer to Figure 5 on page 31). Enabling node quorum with tiebreaker disks starts by designating one or more nodes as quorum nodes. Then one to three disks are defined as tiebreaker disks using the **tiebreakerDisks** parameter on the **mmchconfig** command. You can designate any disk in the file system to be a tiebreaker.

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

Cluster node rules:

1. There is a maximum of eight quorum nodes.
2. All quorum nodes need to have access to all of the tiebreaker disks.
3. You should include the primary and secondary cluster configuration servers as quorum nodes.
4. You may have an unlimited number of non-quorum nodes.
5. If a network connection fails, which causes the loss of quorum, and quorum is maintained by tie-breaker disks, the following rationale is used to re-establish quorum. If a group has the cluster manager, it is the "survivor". The cluster manager can give up its role if it communicates with fewer than the minimum number of quorum nodes as defined by the **minQuorumNodes** configuration parameter. In this case, other groups with the minimum number of quorum nodes (if they exist) can choose a new cluster manager.

Changing quorum semantics:

1. To configure more than eight quorum nodes, you must disable node quorum with tiebreaker disks and restart the GPFS daemon. To disable node quorum with tiebreaker disks:
   a. Issue the **mmshutdown -a** command to shut down GPFS on all nodes.
   b. Change quorum semantics by issuing **mmchconfig tiebreakerdisks=no**.
   c. Add additional quorum nodes.

d. Issue the **mmstartup -a** command to restart GPFS on all nodes.

2. If you remove quorum nodes and the new configuration has less than eight quorum nodes, you can change the configuration to node quorum with tiebreaker disks. To enable quorum with tiebreaker disks:
   a. Issue the **mmshutdown -a** command to shut down GPFS on all nodes.
   b. Delete the appropriate quorum nodes or run **mmchnode --nonquorum** to drop them to a client.
   c. Change quorum semantics by issuing the **mmchconfig tiebreakerdisks="***diskList***"** command.
      - The *diskList* contains the names of the tiebreaker disks.
      - The list contains the NSD names of the disks, preferably one or three disks, separated by a semicolon (;) and enclosed by quotes.
   d. Issue the **mmstartup -a** command to restart GPFS on all nodes.

Tiebreaker disk rules:

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

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

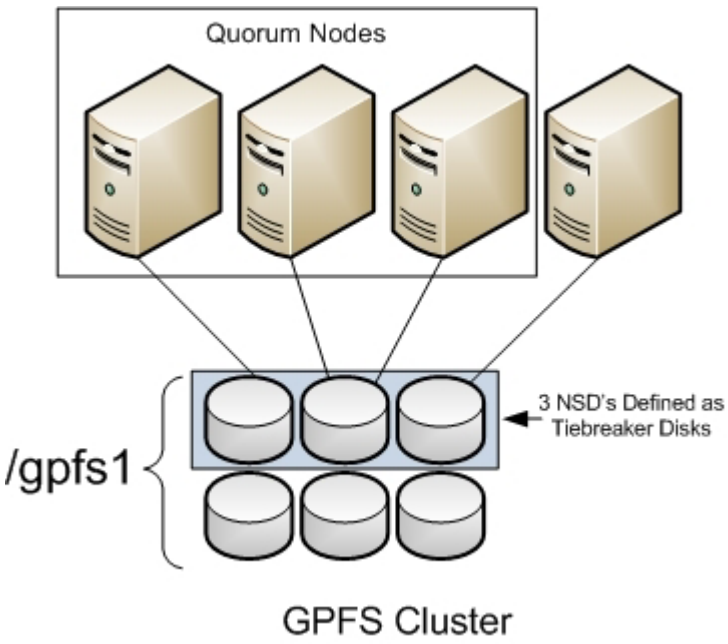

Figure 5. GPFS configuration using node quorum with tiebreaker disks

Selecting quorum nodes

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

- Select nodes that are likely to remain active
  - If a node is likely to be rebooted or require maintenance, do not select that node as a quorum node.
- Select nodes that have different failure points such as:
  - Nodes located in different racks

  - Nodes connected to different power panels
- You should select nodes that GPFS administrative and serving functions rely on such as:
  - Primary configuration servers
  - Secondary configuration servers
  - Network Shared Disk servers
- Select an odd number of nodes as quorum nodes
  - The suggested maximum is seven quorum nodes.
- Having a large number of quorum nodes may increase the time required for startup and failure recovery.
  - Having more than seven quorum nodes does not guarantee higher availability.

Network Shared Disk server and disk failure

The three most common reasons why data becomes unavailable are:

- Disk failure
- Disk server failure with no redundancy
- Failure of a path to the disk

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

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

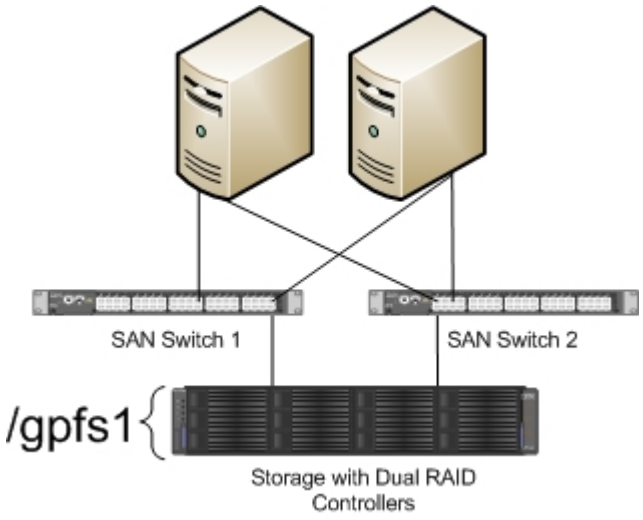

Figure 6. An example of a highly available SAN configuration for a GPFS file system

- Utilizing the GPFS data and metadata replication features (see "Increased data availability" on page 3) along with the designation of failure groups (see "Network Shared Disk (NSD) creation considerations" on page 40 and Figure 7 on page 33)

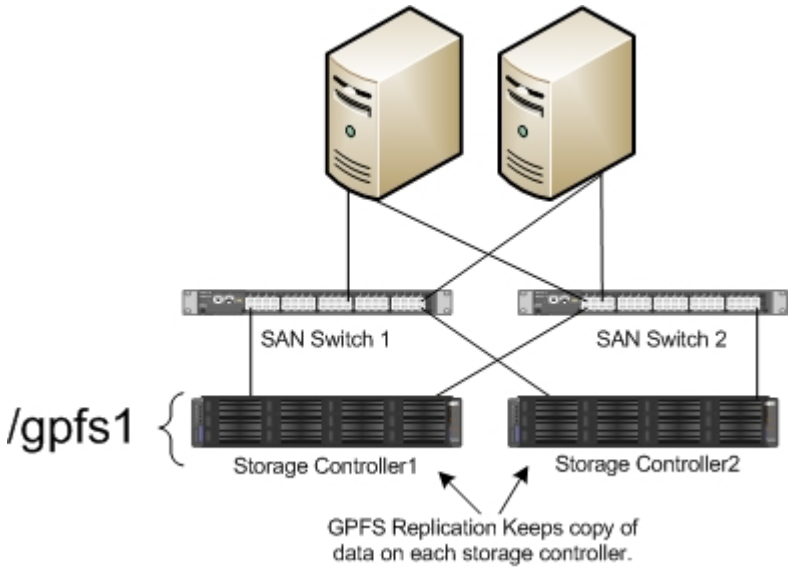

Figure 7. Configuration using GPFS replication for improved availability

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

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

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

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

- Creating multiple NSD servers for all disks. As GPFS determines the available connections to disks in the file system, it is recommended that you always define more than one NSD server for each disk. GPFS allows you to define up to eight NSD servers for each NSD. In a SAN configuration where NSD servers have also been defined, if the physical connection is broken, GPFS dynamically switches to the next available NSD server (as defined on the server list) and continues to provide data. When GPFS discovers that the path has been repaired, it moves back to local disk access. This is the default behavior, which can be changed by designating file system mount options. For example, if you never want a node to use the NSD server path to a disk, even if the local path fails, you can set the **-o useNSDserver** mount option to **never**. You can set the mount option using the **mmchfs**, **mmmount**, **mmremotefs**, and **mount** commands.
- Using an I/O driver that provides multiple paths to the disks for failover purposes. Failover is a path-management algorithm that improves the reliability and availability of a device because the system automatically detects when one I/O path fails and reroutes I/O through an alternate path.

Reduced recovery time using Persistent Reserve

Persistent Reserve (PR) provides a mechanism for reducing recovery times from node failures. To enable PR and to obtain recovery performance improvements, your cluster requires a specific environment:

- *All* disks must be PR-capable
- *All* disks must be hdisks
- If the disks have defined NSD servers, all NSD server nodes must be running AIX

- If the disks are SAN-attached to all nodes, all nodes in the cluster must be running AIX

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

GPFS cluster creation considerations

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

Table 3 details:

- How to change the options
- What the default values are for each option

Table 3. GPFS cluster creation options

Cluster option	Command to change the option	Default value
"Nodes in your GPFS cluster" on page 35	Add nodes through the **mmaddnode** command or delete nodes through the **mmdelnode** command	None
Node designation: **manager** or **client**, see "Nodes in your GPFS cluster" on page 35	**mmchnode**	**client**
Node designation: **quorum** or **nonquorum**, see "Nodes in your GPFS cluster" on page 35	**mmchnode**	**nonquorum**
Primary cluster configuration server, see "GPFS cluster configuration servers" on page 36	**mmchcluster**	None
Secondary cluster configuration server, see "GPFS cluster configuration servers" on page 36	**mmchcluster**	None
"Remote shell command" on page 36	**mmchcluster**	**/usr/bin/rsh**
"Remote file copy command" on page 37	**mmchcluster**	**/usr/bin/rcp**
"Cluster name" on page 37	**mmchcluster**	The node name of the primary GPFS cluster configuration server
GPFS administration adapter port name, see "GPFS node adapter interface names" on page 35	**mmchnode**	Same as the GPFS communications adapter port name
GPFS communications adapter port name, see "GPFS node adapter interface names" on page 35	**mmchnode**	None
"User ID domain for the cluster" on page 38	**mmchconfig**	The name of the GPFS cluster
"Starting GPFS automatically" on page 38	**mmchconfig**	**no**
"Cluster configuration file" on page 38	Not applicable	None

GPFS node adapter interface names

An adapter interface name refers to the hostname or IP address that GPFS uses to communicate with a node. Specifically, the hostname or IP address identifies the communications adapter over which the GPFS daemons or GPFS administration commands communicate. The administrator can specify two node adapter interface names for each node in the cluster:

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

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

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

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

Nodes in your GPFS cluster

When you create your GPFS cluster, you must provide a file containing a list of node descriptors, for each node to be included in the cluster. The node descriptors can be included in the command line, or they can be contained in a separate node descriptor file with one node definition per line. Each descriptor must be specified in this form:

```
NodeName:NodeDesignations:AdminNodeName
```

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

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

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

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

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

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

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

    In general, it is a good idea to define mord than one node as a manager node. How many nodes you designate as manager depends on the workload and the number of GPFS server licenses you have. If you are running large parallel jobs, you may need more manager nodes than in a four-node cluster supporting a Web application. As a guide, in a large system there should be a different file system manager node for each GPFS file system.

- **quorum** | **nonquorum** – This designation specifies whether or not the node should be included in the pool of nodes from which quorum is derived. The default is **nonquorum**. You need to designate at least one node as a quorum node. It is recommended that you designate at least the primary and secondary cluster configuration servers and NSD servers as quorum nodes.

  How many quorum nodes you designate depends upon whether you use node quorum or node quorum with tiebreaker disks. See "Quorum" on page 29.

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

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

Follow these rules when adding nodes to your GPFS cluster:

- While a node may mount file systems from multiple clusters, the node itself may only reside in a single cluster. Nodes are added to a cluster using the **mmcrcluster** or **mmaddnode** command.
- The nodes must be available when they are added to a cluster. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node and then issue the **mmaddnode** command to add those nodes.
- Designate at least one but not more than seven nodes as quorum nodes. When not using tiebreaker disks, you can designate more quorum nodes, but it is recommended to use fewer than eight if possible. It is recommended that you designate the cluster configuration servers as quorum nodes. How many quorum nodes altogether you will have depends on whether you intend to use the node quorum with tiebreaker algorithm or the regular node based quorum algorithm. For more details, see "Quorum" on page 29.

GPFS cluster configuration servers

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

If you do not specify a secondary cluster configuration server:

1. If your primary server fails, the GPFS cluster configuration data files are inaccessible, which results in the failure of any GPFS administration commands that are issued. Similarly, when the GPFS daemon starts up, at least one of the two GPFS cluster configuration servers must be accessible. See "Cluster configuration data files" on page 24.
2. If the primary server fails, you can use the **mmchcluster** command with the **-p** option to designate another node as the primary cluster configuation server. Similarly, you can use the **mmchcluster** command with the **-s** option to define a secondary cluster configuration server. See the *GPFS: Administration and Programming Reference* for more information about the **mmchcluster** command.

Remote shell command

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

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

Clusters that include both UNIX and Windows nodes must use **ssh** for the remote shell command. For more information, see "Installing and configuring OpenSSH" on page 82.

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

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

For additional information, see the topic that discusses requirements for administering a GPFS file system in the *GPFS: Administration and Programming Reference*.

Remote file copy command

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

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

Clusters that include both UNIX and Windows nodes must use **scp** for the remote copy command. For more information, see "Installing and configuring OpenSSH" on page 82.

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

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

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

Cluster name

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

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

User ID domain for the cluster

The user ID domain for a cluster when accessing a file system remotely. This option is further explained in the *GPFS: Advanced Administration Guide* and the white paper entitled *UID Mapping for GPFS in a Multi-cluster Environment* (http://www-03.ibm.com/systems/clusters/software/whitepapers/uid_gpfs.html).

Starting GPFS automatically

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

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

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

Cluster configuration file

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

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

GPFS license designation

According to the GPFS Licensing Agreement, each node in the cluster must be designated as possessing a GPFS client license or a GPFS server license. The full text of the Licensing Agreement is provided with the installation media and can be found at:

Software license agreements (http://www.ibm.com/software/sla/sladb.nsf).

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

- The GPFS server license permits the licensed node to do the following:
  - Mount GPFS file systems and access data from operating system block devices
  - Perform GPFS management functions such as cluster configuration manager, quorum node, manager node, and NSD server
  - Share GPFS data through any application, service, protocol, or method, such as Network File System (NFS), Common Internet File System (CIFS), File Transfer Protocol (FTP), or Hypertext Transfer Protocol (HTTP)
- The GPFS client license permits a node to do the following:
  - Mount GPFS file systems and access data from operating system block devices and NSD servers
  - Exchange data with other nodes that mount the same file system locally. No other export of the data is permitted

The GPFS license designation is achieved by issuing the appropriate **mmchlicense** command. The number and type of licenses currently in effect for the cluster can be viewed using the **mmlslicense** command. See the *GPFS: Administration and Programming Reference* for more information about these commands.

Disk considerations

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

Infrastructure

- Ensure that you have sufficient disks to meet the expected I/O load. In GPFS terminology, a disk may be a physical disk or a RAID device.
- Ensure that you have sufficient connectivity (adapters and buses) between disks and network shared disk servers.
- Determine whether you are within GPFS limits. Starting with GPFS 3.1, the structural limit on the maximum number of disks in a file system increased from 2048 to 4096; however, the current version of GPFS still enforces the original limit of 2048. Should your environment require support for more than 2048 disks, contact IBM to discuss increasing the enforced limit. (However, the number of disks in your system is often constrained by products other than GPFS.)
- For a list of storage devices tested with GPFS, see the GPFS FAQ at the following URL: **http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.pdf**

Disk access method

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

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

     In this configuration, a single node with connectivity to a disk performs data shipping to all other nodes. This node is the first NSD server specified on the NSD server list. You can define additional NSD servers on the server list. Having multiple NSD servers guards against the loss of a single NSD server. When using multiple NSD servers, all NSD servers must have connectivity to the same disks. In this configuration, all nodes that are not NSD servers will receive their data over the local area network from the first NSD server on the server list. If the first NSD server fails, the next available NSD server on the list will control data distribution.
  3. A combination of SAN-attached and an NSD server configuration.

     Configuration consideration:

     - If the node has a physical attachment to the disk and that connection fails, the node switches to using a specified NSD server to perform I/O. For this reason, it is recommended that you define NSDs with multiple servers, even if all nodes have physical attachments to the disk.
     - Configuring GPFS disks without an NSD server stops the serving of data when the direct path to the disk is lost. This may be a preferable option for nodes requiring a higher speed data connection provided through a SAN as opposed to a lower speed network NSD server connection. Parallel jobs using MPI often have this characteristic.

– The **-o useNSDserver** file system mount option on the **mmmount**, **mount**, **mmchfs**, and **mmremotefs** commands can be used to specify the disk discovery, and limit or eliminate switching from local access to NSD server access, or the other way around.

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

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

  Policies provide for the following:

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

  See the *GPFS: Advanced Administration Guide* for more information.

Disk considerations include:

1. "Network Shared Disk (NSD) creation considerations"
2. "NSD server considerations" on page 42
3. "File system descriptor quorum" on page 43

Network Shared Disk (NSD) creation considerations

You must prepare each physical disk you intend to use with GPFS by first defining it as a Network Shared Disk (NSD) using the **mmcrnsd** command. NSDs can be created on the following types of physical disks:

- An hdisk or vpath on AIX
- A block disk device or a disk partition on Linux
- A disk drive on Windows

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

The **mmcrnsd** 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:

```
DiskName:ServerList::DiskUsage:FailureGroup:DesiredName:StoragePool
```

DiskName

On UNIX, the block device name that appears in **/dev** for the disk you want to define as an NSD. Examples of disks that are accessible through a block device are directly attached disks. GPFS disk discovery looks for common device types, **/dev/sd** on Linux for example. If you are using nonstandard devices or you are not sure, see "NSD disk discovery" on page 23. If a *ServerList* is specified, *DiskName* must be the **/dev** name for the disk device on the first NSD server node defined in the server list. This is due to the fact that when the **mmcrnsd** command is executed, the NSD descriptors are written by the first node in the *ServerList* parameter.

On Windows, the disk number (for example, 3) of the disk you want to define as an NSD. Disk numbers appear in Windows Disk Management console and the DISKPART command line utility. If a server node is specified, *DiskName* must be the disk number from the first NSD server node defined in the server list.

See the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for the latest supported disk types.

ServerList

A comma-separated list of NSD server nodes having the form:

```
server1[,server2,...,server8]
```

You can specify up to eight NSD servers in this list. A GPFS node accessing the disk through an NSD server uses the first server on the list with which it can communicate. If the first server is not available, the node uses the next available server on the list. If you do not define a server list, GPFS assumes that the disk is SAN-attached to all nodes in the cluster. If all nodes in the cluster do not have access to the disk, or if the file system to which the disk belongs is to be accessed by other GPFS clusters, you must specify a server list.

DiskUsage

Specify a disk usage or accept the default. This field is used at file system creation, so it is ignored by the **mmcrnsd** command and is passed unchanged to the output descriptor file produced by **mmcrnsd**. Possible values are:

- **dataAndMetadata** – Indicates that the disk contains both data and metadata. This is the default for the **system** storage pool.
- **dataOnly** – Indicates that the disk contains data and does not contain metadata. This is the default and only allowed value for all storage pools other than the **system** pool.
- **metadataOnly** – Indicates that the disk contains metadata and does not contain data. Only NSDs in the system pool can be designated **metadataOnly**.
- **descOnly** – Indicates that the disk contains no data and no metadata. Such a disk is used solely to keep a copy of the file system descriptor. This disk usage is most commonly used as a third failure group in certain disaster recovery configurations.

FailureGroup

A number identifying the failure group to which this disk belongs. You can specify any value from -1 to 4000 (where -1 indicates that the disk has no point of failure in common with any other disk). If you do not specify a failure group, the value defaults to the node number plus 4000 for the first NSD server defined in the server list. If you do not specify an NSD server, the value defaults to -1. GPFS uses this information during data and metadata placement to ensure 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 NSD server or adapter should be placed in the same failure group.

This field is ignored and passed unchanged to the output descriptor file written by either the **mmcrnsd** command or the **mmcrvsd** command.

DesiredName

Specify the name you desire for the NSD to be created. This name must not already be used as another GPFS disk name, and it must not begin with the reserved string 'gpfs'.

Note: This name can contain only the characters: 'A' through 'Z', 'a' through 'z', '0' through '9', or '_' (the underscore). All other characters are not valid.

If a desired name is not specified, the NSD is assigned a name according to the convention:

gpfs*NN***nsd**

where *NN* is a unique nonnegative integer not used in any prior NSD.

StoragePool

Specifies the name of the storage pool to which the NSD is assigned. Storage pool names:

- Must be unique within a file system, but not across file systems
- Should not be larger than 255 alphanumeric characters
- Are case sensitive:

  For example, “MYpool” and “myPool” are distinct storage pools.

If this name is not provided, the default is **system**. Only the **system** pool may contain **metadataOnly**, **dataAndMetadata**, or **descOnly** disks.

Upon successful completion of the **mmcrnsd** command the disk descriptors in the input file are rewritten:

- The original descriptor line is copied and commented out.
- The physical device name is replaced with the assigned unique global name.
- The NSD server names defined in the server list are omitted.
- The *DiskUsage*, *FailureGroup*, and *StoragePool* fields, when provided by the user, are not changed. If the those values are not provided, the appropriate default values are inserted.

The rewritten disk descriptor file, *DescFile*, can then be used as input to the **mmcrfs**, **mmadddisk**, or the **mmrpldisk** commands. The *Disk Usage* and *FailureGroup* specifications in the disk descriptor are only preserved in the *DescFile* file rewritten by the **mmcrnsd** command. 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 NSD server nodes for an existing NSD can be changed later with the **mmchnsd** command. Similarly the *DiskUsage* and *FailureGroup* values for a disk can be changed with the **mmchdisk** command. *StoragePools* can be changed by deleting a disk and adding it back in with the changed pool name. The global NSD name cannot be changed.

Table 4 details the use of disk descriptor information by the GPFS disk commands:

Table 4. Disk descriptor usage for the GPFS disk commands

	mmcrnsd	**mmchnsd**	**mmchdisk**	**mmcrfs**	**default value**
Disk name	X	X	X	X	none
Server list	X	X	NA	NA	none
Disk usage	X	NA	X	X	dataAndMetadata for system storage pool Otherwise, dataOnly for all other storage pools
Failure group	X	NA	X	X	-1 for disks directly attached to all nodes in the cluster Otherwise, node number plus 4000 for the first NSD server that is defined in the server list
Desired name	X	NA	NA	NA	**gpfs***NN***nsd**, where *NN* is a unique nonnegative integer not used in any prior NSD
Storage pool	X	NA	NA	X	system

Note:

1. X – indicates the option is processed by the command
2. NA (not applicable) – indicates the option is not processed by the command

NSD server considerations

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

- Will your Network Shared Disk servers be dedicated servers or will you also be using them to run applications? If you will have non-dedicated servers, consider running less time-critical applications on

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

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

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

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

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

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

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

File system descriptor quorum

A GPFS structure called the *file system descriptor* is initially written to every disk in the file system and is replicated on a subset of the disks as changes to the file system occur, such as the adding or deleting of disks. Based on the number of failure groups and disks, GPFS creates one to five replicas of the descriptor:

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

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

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

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

The loss of all disks in a disk failure group might cause a majority of file systems descriptors to become unavailable and inhibit further file system operations. For example, if your file system is backed up by three or more disks that are assigned to two separate disk failure groups, one of the failure groups will be assigned two of the file system descriptor replicas, while the other failure group will be assigned only one replica. If all of the disks in the disk failure group that contains the two replicas were to become unavailable, the file system would also become unavailable. To avoid this particular scenario, you might want to introduce a third disk failure group consisting of a single disk that is designated as a **descOnly** disk. This disk would exist solely to contain a replica of the file system descriptor (that is, it would not contain any file system metadata or data). This disk can be as small as 4MB.

For more information on this topic, see “Network Shared Disk (NSD) creation considerations” on page 40 and the topic "Establishing disaster recovery for your GPFS cluster" in the *GPFS: Advanced Administration Guide*.

File system creation considerations

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

Each of these factors can help you to determine how much disk resource to devote to the file system, which block size to choose, where to store data and metadata, and how many replicas to maintain. For the latest supported file system size, see the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

Your GPFS file system is created by issuing the **mmcrfs** command. Table 5 on page 45 details the file system creation options specified on the **mmcrfs** command, which options can be changed later with the **mmchfs** command, and what the default values are.

To move an existing file system into a new GPFS cluster, see *Exporting file system definitions between clusters* in the *GPFS: Administration and Programming Reference*.

Table 5. File system creation options

Options	mmcrfs	mmchfs	Default value
Device name of the file system See "Device name of the file system" on page 47.	X	X	none
-D {**nfs4** | **posix**} semantics for a 'deny-write open lock' See "NFS V4 'deny-write open lock'" on page 48.	X	X	**nfs4**
DiskDesc for each disk in your file system See "Disks for your file system" on page 48.	X	Issue the **mmadddisk** or **mmdeldisk** command to add or delete disks from the file system.	none
-F *DescFile* specifies a file that contains a list of disk descriptors, one per line. See "List of disk descriptors" on page 48.	X	Issue the **mmadddisk** or **mmdeldisk** command to add or delete a file that contains a list of disk descriptors.	none
-A {**yes** | **no** | **automount**} to determine when to mount the file system. See "Deciding how the file system is mounted" on page 48.	X	X	**yes**
-B *BlockSize* to set the data block size: **16K**, **64K**, **128K**, **256K**, **512K**, **1M**, **2M**, or **4M**. See "Block size" on page 48.	X	This value cannot be changed without re-creating the file system.	**256K**
-E {**yes** | **no**} to report exact mtime values. See "mtime values" on page 50.	X	X	**yes**
-j {**cluster** | **scatter**} to determine the block allocation map type. See "Block allocation map" on page 50.	X	NA	See "Block allocation map" on page 50.
-k {**posix** | **nfs4** |**all**} to determine the authorization types supported by the file system. See "File system authorization" on page 50.	X	X	**all**
-K {**no** | **whenpossible** | **always**} to enforce strict replication. See "Strict replication" on page 51.	X	X	**whenpossible**

Table 5. File system creation options (continued)

Options	mmcrfs	mmchfs	Default value
-L *LogFileSize* to specify the size of the internal log file. See "GPFS recovery logs" on page 13.	X	This value cannot be changed.	**4 MB**
-m *DefaultMetadataReplicas* See "File system replication parameters" on page 51.	X	X	**1**
-M *MaxMetadataReplicas* See "File system replication parameters" on page 51.	X	This value cannot be changed.	**2**
-n *NumberOfNodes* that will mount the file system. See "Number of nodes mounting the file system" on page 52.	X	X	**32**
-o *MountOptions* to be passed to the **mount** command. See "Assign mount command options" on page 52.	NA	X	none
-Q {**yes** | **no** } to activate quota. See Activate quotas.	X	X	**no**
-r *DefaultDataReplicas* See "File system replication parameters" on page 51.	X	X	**1**
-R *MaxDataReplicas* See "File system replication parameters" on page 51.	X	This value cannot be changed.	**2**
-S {**yes** | **no**} to suppress periodic updating of atime values. See "atime values" on page 49.	X	X	**no**
-t *DriveLetter* See "Windows drive letter" on page 52.	X	X	none
-T *Mountpoint* See "Mountpoint directory" on page 52.	X	X	**/gpfs/***DeviceName*
-V full | **compat** to change the file system format to the latest level. See "Changing the file system format to the latest level" on page 54	NA	X	none

Table 5. File system creation options (continued)

Options	mmcrfs	mmchfs	Default value
-v {**yes** | **no**} to verify disk usage. See "Verifying disk usage" on page 54.	X	NA	**yes**
-W *NewDeviceName* to assign a new device name to the file system.	NA	X	none
-z **yes** | **no** to enable DMAPI See "Enable DMAPI" on page 54.	X	X	**no**
--version to enable only the file system features that are compatible with the specified release. See "Enabling file system features" on page 55.	X	NA	The default is **3.4.0.0**
--filesetdf | **--nofilesetdf** to specify (when quotas are enforced for a fileset) whether the **df** command will report numbers based on the quotas for the fileset and not for the total file system	X	X	
-- inode-limit *MaxNumInodes* [:*NumInodesToPreallocate*] to determine the maximum number of files in the file system. See "Specifying the maximum number of files that can be created" on page 55.	X	X	file system size/**1** MB
--mount-priority *Priority* to control the order in which the individual file systems are mounted at daemon startup or when one of the **all** keywords is specified on the **mmmount** command	X	X	

Note:

1. X – indicates that the option is available on the command.
2. NA (not applicable) – indicates that the option is not available on the command.

Device name of the file system

File system names must be unique within a GPFS cluster. However, two different clusters can have two distinct file systems with the same name. The device name of the file system does not need to be fully qualified. **fs0** is as acceptable as **/dev/fs0**. The name cannot be the same as an existing entry in **/dev**.

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

List of disk descriptors

When creating or modifying a file system, you can specify a file that contains a list of disk descriptors, one per line. You can use the rewritten *DiskDesc* file created by the **mmcrnsd** command, create your own file, or enter the disk descriptors on the command line. When using the *DiskDesc* file created by the **mmcrnsd** command, the values supplied as input to the command for *Disk Usage* and *FailureGroup* are used. When creating your own file or entering the descriptors on the command line, you must specify these values or accept the system defaults.

NFS V4 'deny-write open lock'

You can specify whether a 'deny-write open lock' blocks writes, which is expected and required by NFS V4 (refer to http://www.nfsv4.org/), Samba, and Windows. See *Managing GPFS access control lists and NFS export* in the *GPFS: Administration and Programming Reference*.

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

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

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

Disks for your file system

Disks must be defined as NSDs before they can be added to a GPFS file system. NSDs are created using the **mmcrnsd** command. You can use the **mmlsnsd -F** command to display a list of available NSDs.

See "Disk considerations" on page 39.

Deciding how the file system is mounted

Specify when the file system is to be mounted:

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

no Manual mount.

automount
When the file system is first accessed.

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

Considerations:

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

Block size

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

GPFS supports these block sizes for file systems: 16 KB, 64 KB, 128 KB, 256 KB, 512 KB, 1 MB, 2 MB and 4 MB. This value should be specified with the character **K** or **M** as appropriate, for example: `512K` or `4M`. You should choose the block size based on the application set that you plan to support and whether you are using RAID hardware:

- The 16 KB block size optimizes the use of disk storage at the expense of large data transfers.
- The 64 KB block size offers a compromise if there are a mix of many files of approximately 64K or less in size. It makes more efficient use of disk space than 256 KB, while allowing faster sequential I/O operations than 16 KB.
- The 256 KB block size is the default block size and normally is the best block size for file systems that have a mixed usage or wide range of file size from very small to large files.
- The 1 MB block size can be more efficient if the dominant I/O pattern is sequential access to large files (1 MB or more).

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

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

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

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

Fragments and subblocks

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

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

atime values

atime is a standard file attribute that represents the time when the file was last accessed. The **-S** file system configuration parameter controls how the **atime** value is updated. The default is **-S no**, which results in updating **atime** locally in memory whenever a file is read, but the value is not visible to other nodes until after the file is closed. If an accurate **atime** value is needed, the application must use the GPFS calls **gpfs_stat()** and **gpfs_fstat()** functions. When **-S yes** is specified, or the file system is mounted **read-only**, the updating of the **atime** value is suppressed. This means that the **atime** value is no longer

updated. This can be an issue if you have policies that use the **ACCESS_TIME** file attribute for file management. For more information, see the topic, *Exceptions to the Open Group technical standards* in the *GPFS: Administration and Programming Reference*.

mtime values

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

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

Block allocation map

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

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

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

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

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

scatter
: GPFS chooses the location of the blocks randomly.

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

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

File system authorization

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

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

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

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

Avoid specifying **nfs4** or **all** unless files will be exported to NFS V4 clients or the file system will be mounted on Windows.

Strict replication

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

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

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

always
Indicates that strict replication is enforced.

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

Internal log file

You can specify the internal log file size. Refer to "GPFS recovery logs" on page 13 for additional information.

File system replication parameters

The metadata (inodes, directories, and indirect blocks) and data replication parameters are set at the file system level and apply to all files. They are initially set for the file system when issuing the **mmcrfs** command. They can be changed for an existing file system using the **mmchfs** command. When the replication parameters are changed, files created after the change are affected. To apply the new replication values to existing files in a file system, issue the **mmrestripefs** command.

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

Default metadata replicas

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

Maximum metadata replicas

The maximum number of copies of metadata for all files in the file system can be specified at file system creation by using the **-M** option on the **mmcrfs** command. The default is 2. The allowable values are 1 or 2, but it cannot be lower than the value of *DefaultMetadataReplicas*. This value cannot be changed.

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.

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

Maximum data replicas

The maximum number of copies of data blocks for a file can be specified at file system creation by using the **-R** option on the **mmcrfs** command. The default is 2. The allowable values are 1 and 2, but cannot be lower than the value of *DefaultDataReplicas*. This value cannot be changed.

Number of nodes mounting the file system

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

When creating a GPFS file system, over-estimate the number of nodes that will mount the file system. This input is used in the creation of GPFS data structures that are essential for achieving the maximum degree of parallelism in file system operations (see Chapter 2, "GPFS architecture," on page 9). Although a larger estimate consumes a bit more memory, insufficient allocation of these data structures can limit the ability to process certain parallel requests efficiently, such as the allotment of disk space to a file. If you cannot predict the number of nodes, allow the default value to be applied. Specify a larger number if you expect to add nodes, but avoid wildly overestimating as this can affect buffer operations.

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

Windows drive letter

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

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

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

Mountpoint directory

Every GPFS file system has a default mount point associated with it. This mount point can be specified and changed with the **-T** option of the **mmcrfs** and **mmchfs** commands. If you do not specify a mount point when you create the file system, GPFS will set the default mount point to **/gpfs/***DeviceName*.

Assign mount command options

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

Enabling quotas

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

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

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

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

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

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

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

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

Default quotas

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

Default quotas may be set for a file system only if the file system was created with the **-Q yes** option on the **mmcrfs** command, 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. Default values are established by issuing the **mmdefedquota** command.

Quota system files

The GPFS quota system maintains three separate files that contain data about usage and limits. These files reside in the root directory of the GPFS file systems when quotas are enabled:

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

- **fileset.quota**

All three **.quota** files are:

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

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

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

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

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.

Note: A file system *cannot* be mounted by Windows nodes if DMAPI is enabled.

For further information about DMAPI for GPFS, see the *GPFS: Data Management API Guide*.

Verifying disk usage

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

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

Changing the file system format to the latest level

You can change the file system format to the latest format supported by the currently-installed level of GPFS by issuing the **mmchfs** command with the **-V full** option or the **-V compat** option. The **full** option enables all new functionality that requires different on-disk data structures. This may cause the file system to become permanently incompatible with earlier releases of GPFS. The **compat** option enables only changes that are backward compatible with the previous GPFS release. If all GPFS nodes that are accessing a file system (both local and remote) are running GPFS 3.4, then it is safe to use **full** option. Certain features may require that **mmmigratefs** be run on the unmounted file system.

Enabling file system features

By default, new file systems are created with all currently available features enabled. Since this may prevent clusters that are running earlier GPFS releases from accessing the file system, you can enable only the file system features that are compatible with the specified release by issuing the **mmcrfs** command with the **--version** *Version* option. See the *GPFS: Administration and Programming Reference* for more information.

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

You can specify (when quotas are enforced for a fileset) whether the **df** command will report numbers based on the quotas for the fileset and not for the total file system. To do so, use the **--filesetdf** | **--nofilesetdf** option on either the **mmchfs** command or the **mmcrfs** command.

Specifying the maximum number of files that can be created

The maximum number of files that can be created can be specified by using the **--inode-limit** option on the **mmchfs** command. Allowable values, which range from the current number of created inodes (determined by issuing the **mmdf** command with the **-F** option) through the maximum number of files that are supported, are constrained by the formula:

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

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

Inodes are allocated when they are used. When a file is deleted, the inode is reused, but inodes are never deallocated. When setting the maximum number of inodes in a file system, there is the option to preallocate inodes. However, in most cases there is no need to preallocate inodes because, by default, inodes are allocated in sets as needed. If you do decide to preallocate inodes, be careful not to preallocate more inodes than will be used; otherwise, the allocated inodes will unnecessarily consume metadata space that cannot be reclaimed.

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

Further considerations when managing inodes:

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

Controlling the order in which file systems are mounted

You can control the order in which the individual file systems are mounted at daemon startup (if **mmlsconfig autotload** shows yes) or when one of the **all** keywords is specified. To do so, use the **--mount-priority** *Priority* option on the **mmcrfs**, **mmchfs**, or **mmremotefs** command.

A sample file system creation

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

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

Enter:

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

The system displays information similar to:

```
The following disks of gpfs2 will be formatted on node k194p03.tes.nnn.com:
    hd25n09: size 17796014 KB
    hd24n09: size 17796014 KB
    hd23n09: size 17796014 KB
Formatting file system ...
Disks up to size 59 GB can be added to storage pool system.
Creating Inode File
  56 % complete on Mon Mar  9 15:10:08 2009
 100 % complete on Mon Mar  9 15:10:11 2009
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
  44 % complete on Mon Mar  9 15:11:32 2009
  90 % complete on Mon Mar  9 15:11:37 2009
 100 % complete on Mon Mar  9 15:11:38 2009
Completed creation of file system /dev/gpfs2.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

To confirm the file system configuration, issue the command:

```
mmlsfs gpfs2
```

The system displays information similar to:

```
| flag             value                    description
| ---------------- ------------------------ -----------------------------------
|  -f              262144                   Minimum fragment size in bytes
|  -i              512                      Inode size in bytes
|  -I              32768                    Indirect block size in bytes
|  -m              2                        Default number of metadata replicas
|  -M              2                        Maximum number of metadata replicas
|  -r              1                        Default number of data replicas
|  -R              2                        Maximum number of data replicas
|  -j              scatter                  Block allocation type
|  -D              nfs4                     File locking semantics in effect
|  -k              all                      ACL semantics in effect
|  -n              32                       Estimated number of nodes that will mount file system
|  -B              262144                   Block size
|  -Q              user;group;fileset       Quotas enforced
|                  user;group               Default quotas enabled
|  -V              12.03 (3.4.0.0)          File system version
|  -u              yes                      Support for large LUNs?
|  -z              no                       Is DMAPI enabled?
|  -L              262144                   Logfile size
|  -E              yes                      Exact mtime mount option
|  -S              yes                      Suppress atime mount option
```

```
|   -K               whenpossible            Strict replica allocation option
|   --create-time    Tue Mar 30 14:56:59 2010 File system creation time
|   --fastea         yes                     Fast external attributes enabled?
|   --filesetdf      no                      Fileset df enabled?
|   --inode-limit    2015232                 Maximum number of inodes
|   -P               system;sp1              Disk storage pools in file system
|   -d               dm13nsd;dm10nsd;dm11nsd  Disks in file system
|   -A               yes                     Automatic mount option
|   -o               none                    Additional mount options
|   -T               /gpfs2                  Default mount point
|   --mount-priority 10                      Mount priority
```

Chapter 4. Steps to establishing and starting your GPFS cluster

There are several steps you must perform to establish and start your GPFS cluster. This topic provides the information you need for performing those steps.

Follow these steps to establish your GPFS cluster:

1. Review supported hardware, software, and limits by reviewing the GPFS FAQ (publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for the latest recommendations on establishing a GPFS cluster.
2. Install the GPFS licensed program on your system:
   - For existing systems, see Chapter 8, "Migration, coexistence and compatibility," on page 85.
   - For new systems:
     - For your Linux nodes, see Chapter 5, "Installing GPFS on Linux nodes," on page 61.
     - For your AIX nodes, see Chapter 6, "Installing GPFS on AIX nodes," on page 67.
     - For your Windows nodes, see Chapter 7, "Installing GPFS on Windows nodes," on page 71.
3. Decide which nodes in your system will be quorum nodes (see "Quorum" on page 29).
4. Create your GPFS cluster by issuing the **mmcrcluster** command. See "GPFS cluster creation considerations" on page 34.
5. Use the **mmchlicense** command to assign an appropriate GPFS license to each of the nodes in the cluster. See "GPFS license designation" on page 38 for more information.

After your GPFS cluster has been established:

1. Ensure you have configured and tuned your system according to the values suggested in Chapter 9, "Configuring and tuning your system for GPFS," on page 91.
2. Start GPFS by issuing the **mmstartup** command. See the *GPFS: Administration and Programming Reference*.
3. Create new disks for use in your file systems by issuing the **mmcrnsd** command. See "Network Shared Disk (NSD) creation considerations" on page 40.
4. Create new file systems by issuing the **mmcrfs** command. See "File system creation considerations" on page 44.
5. Mount your file systems.
6. As an optional step, you can also create a temporary directory (**/tmp/mmfs**) to collect problem determination data. The **/tmp/mmfs** directory can be a symbolic link to another location if more space can be found there. If you decide to do so, the temporary directory should *not* be placed in a GPFS file system, as it might not be available if GPFS fails.

   If a problem should occur, GPFS might write 200 MB or more of problem determination data into **/tmp/mmfs**. These files must be manually removed when any problem determination is complete. This should be done promptly so that a **NOSPACE** condition is not encountered if another failure occurs. An alternate path can be specified by issuing the **mmchconfig dataStructureDump** command.

Chapter 5. Installing GPFS on Linux nodes

There are three steps to installing GPFS on Linux nodes: Preparing the environment; installing the GPFS software; and building the GPFS portability layer. The information in this topic points you to the detailed steps.

Before installing GPFS, you should review Chapter 3, "Planning for GPFS," on page 27 and the GPFS FAQ at the following URL: **http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html**

Installing GPFS without ensuring that the prerequisites listed in "Hardware requirements" on page 27 and "Software requirements" on page 28 are satisfied can lead to undesired results.

The installation process includes:

1. "Preparing the environment"
2. "Installing GPFS software on Linux nodes" on page 62
3. "Building the GPFS portability layer" on page 64

Preparing the environment

Before proceeding with installation, prepare your environment by following the suggestions in the following sections.

Add the GPFS bin directory to your shell PATH

Ensure that the PATH environment variable for the root user on each node includes **/usr/lpp/mmfs/bin**. (This is not required for the operation of GPFS, but it can simplify administration.)

Other suggestions for cluster administration

GPFS commands operate on all nodes required to perform tasks. When you are administering a cluster, it may be useful to have a more general form of running commands on all of the nodes. One suggested way to do this is to use an OS utility like **dsh** or **pdsh** that can execute commands on all nodes in the cluster. For example, you can use **dsh** to check the kernel version of each node in your cluster:

```
# dsh uname -opr
Node01: 2.6.18-128.1.14.e15 x86_64 GNU/Linux
Node02: 2.6.18-128.1.14.e15 x86_64 GNU/Linux
```

Once you have **dsh** set up, you can use it to install GPFS on a large cluster. For details about setting up **dsh** or a similar utility, review the documentation for the utility.

Verify that prerequisite software is installed

Before installing GPFS, it is necessary to verify that you have the correct levels of the prerequisite software installed on each node in the cluster. If the correct level of prerequisite software is *not* installed, see the appropriate installation manual before proceeding with your GPFS installation.

For the most up-to-date list of prerequisite software, see the GPFS FAQ at the following URL: **http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html**

The FAQ contains the latest information about the following:

- Supported Linux distributions and kernel levels
- Recommended or required RPM levels
- Software recommendations
- Configuration information

Installing GPFS software on Linux nodes

Follow the steps in this topic in the specified order to install the GPFS software using the **rpm** command.

This procedure installs GPFS on one node at a time:

1. "Accepting the electronic license agreement"
2. "Extracting the GPFS software"
3. "Extracting GPFS patches (update RPMs)" on page 63
4. "Installing the GPFS man pages" on page 63
5. "Installing the GPFS software" on page 64
6. "Verifying the GPFS installation" on page 64

Accepting the electronic license agreement

The GPFS software license agreement is shipped with the GPFS software and is viewable electronically. When you extract the GPFS software, you are asked whether or not you accept the license. The electronic license agreement must be accepted before software installation can continue. Read the software agreement carefully before accepting the license. See "Extracting the GPFS software."

Extracting the GPFS software

The GPFS software is delivered in a self-extracting archive. The self-extracting image contains the following:

- The GPFS product installation RPMs
- The License Acceptance Process (LAP) Tool

  The LAP Tool is invoked for acceptance of the GPFS license agreements. The license agreements must be accepted to obtain access to the GPFS product installation images.
- A version of the Java™ Runtime Environment (JRE) necessary to run the LAP Tool

Before installing GPFS, you need to extract the RPMs from the archive.

1. Copy the self-extracting product image, **gpfs_install-3.4***, from the CD-ROM to a local directory (where * is the correct version of the product for your hardware platform and Linux distribution).

   Example:

   ```
   cp /media/cdrom/gpfs_install-3.4.0-0_x86_64 /tmp/gpfs_install-3.4.0-0_x86_64
   ```
2. Verify that the self-extracting program has executable permissions:

   ```
   # ls -l /tmp/gpfs_install-3.4.0-0_x86_64
   -rwxr-xr-x 1 root root 110885866 Apr 27 15:52 /tmp/gpfs_install-3.4.0-0_x86_64
   ```
3. Invoke the self-extracting image that you copied from the CD-ROM and accept the license agreement:
   a. By default, the LAP Tool, JRE and GPFS installation images are extracted to the target directory **/usr/lpp/mmfs/3.4*** .
   b. The license agreement files on the media can be viewed in graphics mode or text-only mode.
      1) Graphics mode is the default behavior. To view the files in graphics mode, invoke **gpfs_install-3.4***. Using the graphics-mode installation requires a window manager to be configured.

2) To view the files in text-only mode, add the **--text-only** option. When run in text-only mode, the output explains how to accept the agreement:

```
<...Last few lines of output...>
Press Enter to continue viewing the license agreement, or
enter "1" to accept the agreement, "2" to decline it, "3"
to print it, "4" to read non-IBM terms, or "99" to go back
to the previous screen.
```

c. You can use the **--silent** option to accept the license agreement automatically.

d. Use the **--help** option to obtain usage information from the self-extracting archive.

The following is an example of how to extract the software using text mode:

```
/tmp/gpfs_install_3.4.0-0_x86_64 --text-only
```

Upon license agreement acceptance, the GPFS product installation images are placed in the extraction target directory (**/usr/lpp/mmfs/3.4***). This directory contains the following GPFS RPMs:

- gpfs.base-3.4*.rpm
- gpfs.docs-3.4*.noarch.rpm
- gpfs.gpl-3.4*.noarch.rpm
- gpfs.msg.en_US-3.4*.noarch.rpm

In this directory there is a license subdirectory that contains license agreements in multiple languages:

```
# ls /usr/lpp/mmfs/3.3/license
Chinese_TW.txt French.txt  Japanse.txt         Notices.txt   Slovenian.txt
Chinese.txt    German.txt  Korean.txt          Polish.txt    Spanish.txt
Czech.txt      Greek.txt   Lithuanian.txt      Portguese.txt Status.dat
English.txt    Italian.txt non_ibm_license.txt Russian.txt   Turkish.txt
```

The license agreement remains available in the extraction target directory under the **license** subdirectory for future access. The license files are written using operating system-specific code pages. This enables you to view the license in English and in the local language configured on your machine. The other languages are not guaranteed to be viewable.

Extracting GPFS patches (update RPMs)

Typically when you install a GPFS system there are patches available. It is recommended that you always look for the latest patches when installing or updating a GPFS node.

GPFS patches (update RPMs) are available from the following IBM Web site:

http://www14.software.ibm.com/webapp/set2/sas/f/gpfs/home.html

The update RPMs are distributed in a different form from that of the base software; they are stored in a tar file. Use the **tar** command to extract the update RPMs into a local directory.

Recommendation: Because you need to install the base software RPMs completely before installing a patch level, it is recommended that you place update RPMs in a separate directory from the base RPMs. This enables you to simplify the installation process by using the **rpm** command with wildcards (**rpm -ivh *.rpm**), first on the directory containing the base RPMs, then on the directory containing the update RPMs. (There is no license acceptance required on patches, so once you have extracted the contents of the tar file, the update RPMs are available for installation.)

If you are applying a patch during the initial installation of GPFS on a node, you only need to build the portability layer once after the base and update RPMs are installed.

Installing the GPFS man pages

In order to use the GPFS man pages, the **gpfs.docs** RPM must be installed. Once you have installed the **gpfs.docs** RPM, the GPFS manual pages are located at **/usr/share/man/**.

You do not need to install the **gpfs.docs** RPM on all nodes if man pages are not desired (for example, if local file system space on the node is minimal).

Installing the GPFS software

The GPFS software is installed using the **rpm** command.

Required packages

The following packages are required:

gpfs.base-3.4*.rpm
gpfs.gpl-3.4*.noarch.rpm
gpfs.msg.en_US-3.4*.noarch.rpm

Optional packages

The following package is optional: **gpfs.docs-3.4*noarch.rpm**.

To install all of the GPFS RPMs, issue the following command:

```
rpm -ivh /usr/lpp/mmfs/3.4/gpfs*.rpm
```

Verifying the GPFS installation

You can verify the installation of the GPFS RPMs on each node. To check that the software has been successfully installed, use the **rpm** command:

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

The system should return output similar to the following:

```
gpfs.docs-3.4.0-0
gpfs.base-3.4.0-0
gpfs.msg.en_US-3.4.0-0
gpfs.gpl-3.4.0-0
```

Building the GPFS portability layer

Before starting GPFS, you must build and install the GPFS portability layer.

The GPFS portability layer is a loadable kernel module that allows the GPFS daemon to interact with the operating system.

Note: The GPFS kernel module should be updated any time the Linux kernel is updated or patched. Updating the GPFS kernel module after a Linux kernel update requires rebuilding and installing a new version of the module.

1. Before building the portability layer, check for the following:
   - Updates to the portability layer at the following URL: **http://www14.software.ibm.com/webapp/set2/sas/f/gpfs/home.html**
   - The latest kernel level support in the GPFS FAQ at the following URL: **http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html**
   - Any applicable GPFS Linux kernel patches available at the following URL:

     http://www.ibm.com/developerworks/opensource/

     under the project *General Parallel File System (GPFS) for Linux Kernel Patches*.
2. Build your GPFS portability layer in one of two ways:
   - Using the Autoconfig tool (recommended)
   - Using the directions in **/usr/lpp/mmfs/src/README**

Using the automatic configuration tool to build the GPFS portability layer

To simplify the build process, GPFS provides an automatic configuration tool.

The following example shows the commands required to build the GPFS portability layer using the automatic configuration option (**make Autoconfig**):

```
cd /usr/lpp/mmfs/src
make Autoconfig
make World
make InstallImages
```

Each kernel module is specific to a Linux version and platform. If you have multiple nodes running exactly the same operating system level on the same platform, you can build the kernel module on one node, then create an RPM that contains the binary module for ease of distribution.

If you choose to generate an RPM package for portability layer binaries, perform the following additional step:

```
make rpm
```

When the command finishes, it displays the location of the generated RPM:

```
<...Last line of output...>
Wrote:  /usr/src/redhat/RPMS/x86_64/gpfs.gplbin-2.6.18-128.1.14.e15-3.3.0-1.x86_64.rpm
```

You can then copy the generated RPM package to other machines for deployment. The generated RPM can *only* be deployed to machines with identical architecture, distribution level, Linux kernel, and GPFS maintenance level.

Note: During the package generation, temporary files are written to the **/tmp/rpm** directory, so be sure there is sufficient space available. By default, the generated RPM goes to **/usr/src/packages/RPMS/<arch>** for SUSE Linux Enterprise Server and **/usr/src/redhat/RPMS/<arch>** for Red Hat Enterprise Linux.

Chapter 6. Installing GPFS on AIX nodes

There are three steps to installing GPFS on AIX nodes. The information in this topic will point you to the detailed steps.

Before you begin installation, read Chapter 3, "Planning for GPFS," on page 27 and the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

Do not attempt to install GPFS if you do not have the prerequisites listed in "Hardware requirements" on page 27 and "Software requirements" on page 28.

Ensure that the **PATH** environment variable on each node includes **/usr/lpp/mmfs/bin**.

The installation process includes:

1. "Creating a file to ease the AIX installation process"
2. "Verifying the level of prerequisite software"
3. "Procedure for installing GPFS on AIX nodes" on page 68

Creating a file to ease the AIX installation process

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

For example, create the file **/tmp/gpfs.allnodes**, listing the nodes one per line:

```
k145n01.dpd.ibm.com
k145n02.dpd.ibm.com
k145n03.dpd.ibm.com
k145n04.dpd.ibm.com
k145n05.dpd.ibm.com
k145n06.dpd.ibm.com
k145n07.dpd.ibm.com
k145n08.dpd.ibm.com
```

Verifying the level of prerequisite software

Before you can install GPFS, verify that your system has the correct software levels installed.

If your system *does not* have the prerequisite AIX level, refer to the appropriate installation manual before proceeding with your GPFS installation. See the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for the latest software levels.

To verify the software version, run the command:

```
WCOLL=/tmp/gpfs.allnodes dsh "oslevel"
```

The system should display output similar to:

```
6.1.0.0
```

Procedure for installing GPFS on AIX nodes

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

Follow these steps to install the GPFS software using the **installp** command:

1. "Accepting the electronic license agreement"
2. "Creating the GPFS directory"
3. "Creating the GPFS installation table of contents file"
4. "Installing the GPFS man pages"
5. "Installing GPFS over a network" on page 69
6. "Reconciling existing GPFS files" on page 69
7. "Verifying the GPFS installation" on page 69

Accepting the electronic license agreement

The GPFS software license agreements is shipped and viewable electronically. The electronic license agreement 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.

Creating the GPFS directory

To create the GPFS directory:

1. On any node create a temporary subdirectory where GPFS installation images will be extracted. For example:

   ```
   mkdir  /tmp/gpfslpp
   ```

2. Copy the installation images from the CD-ROM to the new directory, by issuing:

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

   This command places these GPFS installation files in the images directory:

   a. gpfs.base
   b. gpfs.docs.data
   c. gpfs.msg.en_US

Creating the GPFS installation table of contents file

To create the GPFS installation table of contents file:

1. Make the new image directory the current directory:

   ```
   cd /tmp/gpfslpp
   ```

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

   ```
   inutoc .
   ```

Installing the GPFS man pages

In order to use the GPFS man pages you must install the **gpfs.docs.data** image. The GPFS manual pages will be located at **/usr/share/man/**.

Installation consideration: The **gpfs.docs.data** image need not be installed on all nodes if man pages are not desired or local file system space on the node is minimal.

Installing GPFS over a network

Install GPFS according to these directions, where *localNode* is the name of the node on which you are running:

1. If you are installing on a shared file system network, ensure the directory where the GPFS images can be found is NFS exported to all of the nodes planned for your GPFS cluster (**/tmp/gpfs.allnodes**).
2. Ensure an acceptable directory or mountpoint is available on each target node, such as **/tmp/gpfslpp**. If there is not, create one:

   ```
   WCOLL=/tmp/gpfs.allnodes dsh "mkdir /tmp/gpfslpp"
   ```

3. If you are installing on a shared file system network, to place the GPFS images on each node in your network, issue:

   ```
   WCOLL=/tmp/gpfs.allnodes dsh "mount localNode:/tmp/gpfslpp /tmp/gpfslpp"
   ```

   Otherwise, issue:

   ```
   WCOLL=/tmp/gpfs.allnodes dsh "rcp localNode:/tmp/gpfslpp/gpfs* /tmp/gpfslpp"
   WCOLL=/tmp/gpfs.allnodes dsh "rcp localNode:/tmp/gpfslpp/.toc /tmp/gpfslpp"
   ```

4. Install GPFS on each node:

   ```
   WCOLL=/tmp/gpfs.allnodes dsh "installp -agXYd /tmp/gpfslpp gpfs"
   ```

Reconciling existing GPFS files

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/gpfsrecover.src
  /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**.

Verifying the GPFS installation

Verify that the installation procedure placed the required GPFS files on each node by running the **lslpp** command on *each* node:

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

The system should return output similar to:

```
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  gpfs.base                  3.4.0.0  COMMITTED  GPFS File Manager
  gpfs.msg.en_US             3.4.0.0  COMMITTED  GPFS Server Messages - U.S.
                                                 English
```

```
| Path: /etc/objrepos
|   gpfs.base                 3.4.0.0  COMMITTED  GPFS File Manager
|
| Path: /usr/share/lib/objrepos
|   gpfs.docs.data            3.4.0.0  COMMITTED  GPFS Server Manpages and
|                                                 Documentation
```

Note: The path returned by `lslpp -l` shows the location of the package control data used by **installp**. The listed path does not show GPFS file locations. To view GPFS file locations, use the `-f` flag.

Chapter 7. Installing GPFS on Windows nodes

There are several steps to installing GPFS on Windows nodes. The information in this topic will point you to the detailed steps.

Before you begin installation, read the following:

- Chapter 3, "Planning for GPFS," on page 27
- The GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html)
- "GPFS for Windows overview" and all of its subtopics

Do not install GPFS unless you have the prerequisites listed in "Hardware requirements" on page 27 and "Software requirements" on page 28.

The installation process includes:

1. "Installing GPFS prerequisites" on page 75
2. "Procedure for installing GPFS on Windows nodes" on page 79
3. "Configuring a mixed Windows and UNIX cluster" on page 80

To install GPFS for Windows, first configure your Windows systems as described in "Installing GPFS prerequisites" on page 75. This includes steps such as joining an Active Directory domain and installing the Windows Subsystem for UNIX-based Applications (SUA). GPFS installation is simple once the prerequisites are completed. Finally, if your system will be part of a cluster that includes UNIX nodes, follow the steps described in "Configuring a mixed Windows and UNIX cluster" on page 80. This includes creating the GPFS Administration service, installing OpenSSH, and other requirements. Complete this process before performing configuration steps common to all GPFS supported platforms.

Note: For Windows Server 2003 R2 installation instructions and other support details, see the GPFS 3.2.1 edition of *GPFS: Concepts, Planning, and Installation Guide* at the following URL: `http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp`

Note: Throughout this information, UNIX file name conventions are used. For example, the GPFS cluster configuration data is stored in the **/var/mmfs/gen/mmsdrfs** file. On Windows, the UNIX name space starts under the **%SystemRoot%\SUA** directory, and UNIX-style file names need to be converted accordingly. For example, the cluster configuration file mentioned above is **C:\Windows\SUA\var\mmfs\gen\mmsdrfs**.

GPFS for Windows overview

GPFS for Windows participates in a new or existing GPFS 3.4 or GPFS 3.3 cluster in conjunction with AIX and Linux systems. Support includes the following:

- Access to GPFS 3.4 or 3.3 file systems
- User identity mapping between Windows and UNIX
- Windows file system semantics
- Core GPFS parallel data services
- A broad complement of advanced GPFS features

Identity mapping between Windows and UNIX user accounts is a key feature. System administrators can explicitly match users and groups defined on UNIX with those defined on Windows. Users can maintain

file ownership and access rights from either platform. System administrators are not required to define an identity map. GPFS automatically creates a mapping when one is not defined.

GPFS supports the unique semantic requirements posed by Windows. These requirements include case-insensitive names, NTFS-like file attributes, and Windows file locking. GPFS provides a bridge between a Windows and POSIX view of files, while not adversely affecting the functions provided on AIX and Linux.

GPFS for Windows provides the same core services to parallel and serial applications as are available on AIX and Linux. GPFS gives parallel applications simultaneous access to files from any node that has GPFS mounted, while managing a high level of control over all file system operations. System administrators and users have a consistent command interface on AIX, Linux, and Windows. With few exceptions, the commands supported on Windows are identical to those on other GPFS platforms. See "GPFS support for Windows" for a list of commands that Windows clients do not support.

GPFS support for Windows

GPFS supports the Microsoft®'s Windows Server operating systems starting with Windows Server 2008. Limited support for Windows Server 2003 R2 is available in earlier versions of GPFS.

GPFS 3.4 supports all editions of the following operating systems:

- Windows Server 2008 R2
- Windows Server 2008 x64

Limited GPFS support for the Windows platform first appeared in GPFS 3.2.1. Subsequent GPFS releases have expanded the set of available functions and features so that now most GPFS capabilities supported on the UNIX platforms are also supported on Windows.

Changes in GPFS 3.4 that relate to Windows

The following are changes in GPFS 3.4 that relate to Windows:

- Support for Windows Server 2008 R2
- Support for Windows clusters that do not require UNIX nodes
- Support for direct access to disks and for operating as an NSD server
- Support for native Windows authentication in Windows clusters

GPFS limitations on Windows

GPFS for Windows does not fully support all of the GPFS features that are available on AIX and Linux. Some of these limitations constrain how you can configure a GPFS cluster when it includes Windows nodes. The remaining limitations only pertain to Windows nodes rather than the whole cluster.

GPFS for Windows imposes some constraints on how you can configure and operate a cluster when it includes Windows nodes. The following is a list of these limitations:

- File systems must be created with GPFS 3.2.1.5 or higher. Windows nodes can only mount file systems that were formatted with GPFS versions starting with 3.2.1.5. There is no support for upgrading existing file systems that were created with GPFS 3.1 or earlier.
- File systems cannot be DMAPI-enabled. DMAPI-enabled file systems will not mount on a Windows node.

The remaining GPFS for Windows limitations only pertain to the Windows nodes in a cluster:

- The following GPFS commands are not supported on Windows:
  - **mmapplypolicy**

- **mmbackup**
- **mmbackupconfig**, **mmrestoreconfig**
- **mmcheckquota**, **mmdefedquota**, **mmedquota**, **mmlsquota**, **mmrepquota**
- **mmdelacl**, **mmeditacl**, **mmgetacl**, **mmputacl**
- **mmpmon**

- The GPFS Application Programming Interfaces (APIs) are not supported on Windows.
- The native Windows backup utility is not supported.
- Symbolic links that are created on UNIX-based nodes are specially handled by GPFS Windows nodes; they appear as regular files with a size of 0 and their contents cannot be accessed or modified.
- GPFS on Windows nodes attempts to preserve data integrity between memory-mapped I/O and other forms of I/O on the same computation node. However, if the same file is memory mapped on more than one Windows node, data coherency is not guaranteed between the memory-mapped sections on these multiple nodes. In other words, GPFS on Windows does not provide distributed shared memory semantics. Therefore, applications that require data coherency between memory-mapped files on more than one node might not function as expected.

File system name considerations

GPFS file system names should be no longer than 31 characters if the file system will be mounted on a Windows node. GPFS file system names are used as Windows file system labels, and Windows limits the length of these labels to 31 characters. Any attempt to mount a file system with a long name will fail.

You can rename a file system using a command like the following:

```
mmchfs gpfs_file_system_name_that_is_too_long -W gpfs_name_1
```

File name considerations

File names created on UNIX-based GPFS nodes that are not valid on Windows are transformed into valid short names. A name may not be valid either because it is a reserved name like NUL or COM2, or because it has a disallowed character like colon (:) or question mark (?). See the MSDN Library description of File Names, Paths, and Namespaces (http://msdn.microsoft.com/en-us/library/aa365247(VS.85).aspx) for complete details.

Windows applications can use short names to access GPFS files with Windows file names that are not valid. GPFS generates unique short names using an internal algorithm. You can view these short names using **dir /x** in a command prompt.

Table 6 shows an example:

Table 6. Generating short names for Windows

UNIX	Windows
foo+bar.foobar	FO~23Q_Z.foo
foo|bar.-bar	FO~TD}C5._ba
f|bar.-bary	F_AMJ5!._ba

Case sensitivity

Native GPFS is case-sensitive; however, Windows applications can choose to use case-sensitive or case-insensitive names. This means that case-sensitive applications, such as those using Windows support for POSIX interfaces, behave as expected. Native Win32 applications (such as Windows Explorer) have only case-aware semantics.

The case specified when a file is created is preserved, but in general, file names are case insensitive. For example, Windows Explorer allows you to create a file named **Hello.c**, but an attempt to create **hello.c** in the same folder will fail because the file already exists. If a Windows node accesses a folder that contains two files that are created on a UNIX node with names that differ only in case, Windows inability to distinguish between the two files might lead to unpredictable results.

Antivirus software

If more than one GPFS Windows node is running antivirus software that scans directories and files, shared files only need to be scanned by one GPFS node. It is not necessary to scan shared files more than once. When you run antivirus scans from more than one node, schedule the scans to run at different times to allow better performance of each scan, as well as to avoid any conflicts that might arise because of concurrent exclusive access attempts by the antivirus software from multiple nodes. Note that enabling real-time antivirus protection for GPFS volumes could significantly degrade GPFS performance and cause excessive resource consumption.

Tip: Consider using a single, designated Windows node to perform all virus scans.

Differences between GPFS and NTFS

GPFS differs from the Microsoft Windows NT® File System (NTFS) in its degree of integration into the Windows administrative environment, Windows Explorer, and the desktop. The differences are as follows:

- Manual refreshes are required to see any updates to the GPFS namespace.
- You cannot use the recycle bin.
- You cannot use distributed link tracking. This is a technique through which shell shortcuts and OLE links continue to work after the target file is renamed or moved. Distributed link tracking can help you locate the link sources in case the link source is renamed or moved to another folder on the same or different volume on the same computer, or moved to a folder on any computer in the same domain.
- You cannot use NTFS change journaling. This also means that you cannot use the Microsoft Indexing Service or Windows Search Service to index and search files and folders on GPFS file systems.

GPFS does not support the following NTFS features:

- File compression (on individual files or on all files within a folder)
- Encrypted files and directories
- Quota management (GPFS quotas are administered through GPFS-specific commands)
- Reparse points
- Defragmentation and error-checking tools
- Alternate data streams
- The assignment of an access control list (ACL) for the entire drive
- A change journal for file activity
- The scanning of all files or directories that a particular SID owns (**FSCTL_FIND_FILES_BY_SID**)
- Generation of AUDIT and ALARM events specified in a System Access Control List (SACL). GPFS is capable of storing SACL content, but will not interpret it.
- Windows sparse files API
- Transactional NTFS (also known as TxF)

Access control on GPFS file systems

GPFS provides support for the Windows access control model for file system objects.

Each GPFS file or directory has a Security Descriptor (SD) object associated with it and you can use the standard Windows interfaces for viewing and changing access permissions and object ownership (for example, Windows Explorer Security dialog panel). Internally, a Windows SD is converted to an NFS V4 access control list (ACL) object, which ensures that access control is performed consistently on other supported operating systems. GPFS supports all discretionary access control list (DACL) operations, including inheritance. GPFS is capable of storing system access control list (SACL) objects, but generation of AUDIT and ALARM events specified in SACL contents is not supported.

An important distinction between GPFS and Microsoft Windows NT File Systems (NTFS) is the default set of permissions for the root (top-level) directory on the file system. On a typical NTFS volume, the DACL for the top-level folder has several inheritable entries that grant full access to certain special accounts, as well as some level of access to nonprivileged users. For example, on a typical NTFS volume, the members of the local group **Users** would be able to create folders and files in the top-level folder. This approach differs substantially from the traditional UNIX convention where the root directory on any file system is only writable by the local **root** superuser by default. GPFS adheres to the latter convention; the root directory on a new file system is only writable by the UNIX user **root**, and does not have an extended ACL when the file system is created. This is to avoid impacting performance in UNIX-only environments, where the use of extended ACLs is not common.

When a new GPFS file system is accessed from a Windows client for the first time, a security descriptor object is created for the root directory automatically, and it will contain a noninheritable DACL that grants full access to the local **Administrators** group and read-only access to the local **Everyone** group. This allows only privileged Windows users to create new files and folders. Because the root directory DACL has no inheritable entries, new objects will be created with a default DACL that grants local **Administrators** and **SYSTEM** accounts full access. Optionally, the local system administrator could create a subdirectory structure for Windows users, and explicitly set DACLs on new directories as appropriate (for example, giving the necessary level of access to nonprivileged users).

Note: Some applications expect to find NTFS-style permissions on all file systems and they might not function properly when that is not the case. Running such an application in a GPFS folder where permissions have been set similar to NTFS defaults might correct this.

Installing GPFS prerequisites

This topic provides details on configuring Windows systems prior to installing GPFS.

Perform the following steps:

1. "Configuring Windows" on page 76
   a. "Assigning a static IP address" on page 76
   b. "Joining an Active Directory domain" on page 76
   c. "Disabling User Account Control" on page 76
   d. "Disabling the Windows firewall" on page 76
   e. "Installing the TraceFmt program" on page 78
2. "Installing the Subsystem for UNIX-based Applications (SUA)" on page 77
   a. "Downloading and installing SUA hotfix updates" on page 78

See the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for the latest:

- Software recommendations
- Configuration information

Perform these steps as a member of the **Administrator** group. Once GPFS is installed, the account you use for GPFS administrative operations (such as creating a cluster and file systems) will depend on your

cluster type. For Windows clusters, run GPFS commands as a member of the **Domain Admins** group. For clusters with both Windows and UNIX nodes, run GPFS commands as **root**, a special domain user account described in "Creating the GPFS administrative account" on page 81.

For additional requirements when setting up clusters that contain both Windows nodes and AIX or Linux nodes, see "Configuring a mixed Windows and UNIX cluster" on page 80.

Configuring Windows

This topic provides some details on installing and configuring Windows on systems that will be added to a GPFS cluster.

GPFS 3.4 and GPFS 3.3 support all editions of Windows Server 2008 R2 and Windows Server 2008 for x64-based systems, including Windows HPC Server 2008. GPFS 3.2.1 supports Microsoft Windows Server 2003 R2 for x64-based systems as a GPFS client node.

Assigning a static IP address

GPFS communication requires invariant static IP addresses for each GPFS node.

Joining an Active Directory domain

All Windows systems in the same GPFS cluster should be members of the same Active Directory domain. Join these systems to the Windows domain before adding them to a GPFS cluster.

GPFS expects that all Windows nodes in a cluster are members of the same domain. This gives domain users a consistent identity and consistent file access rights independent of the system they are using. The domain controllers, which run the Active Directory Domain Services, are not required to be members of the GPFS cluster.

Refer to your Windows Server documentation for information on how to install and administer Active Directory Domain Services.

Disabling User Account Control

On Windows Server 2008 nodes, you must disable User Account Control (UAC) for GPFS to operate correctly. UAC needs to be disabled for the entire system, not just turned off for GPFS administrative users.

Windows Server 2008 R2 does not have this requirement. On these systems, GPFS administrative users run in Admin Mode such as when a program is started with the Run as Administrator option.

To disable UAC on Windows Server 2008 systems, follow these steps:

1. Open the **System Configuration** application, which is under **Administrative Tools**.
2. Select the **Tools** tab and scroll down to select **Disable UAC**.
3. Click **Launch**.

Note: This change requires a reboot.

Disabling the Windows firewall

GPFS requires that you modify the default Windows Firewall settings. The simplest change that will allow GPFS to operate properly is to disable the firewall. Open **Windows Firewall** in the Control Panel and click **Turn Windows firewall on or off**, and select **Off** under the General tab. For related information, see the *GPFS port usage* topic in the *GPFS: Advanced Administration Guide*.

Installing the Subsystem for UNIX-based Applications (SUA)

The Subsystem for UNIX-based Applications (SUA) is a POSIX subsystem included with Windows Server. GPFS uses this component to support many of its programs and administrative scripts. System administrators have the option of using either the standard Windows Command Prompt or an SUA shell such as **ksh** to run GPFS commands.

The SUA environment is composed of two parts:

1. The Subsystem for UNIX-based Applications (SUA)
2. The Utilities and SDK for UNIX-based Applications (SUA Utilities and SDK)

Both parts must be installed before installing GPFS. SUA is a feature included with Windows Server and provides runtime support for POSIX applications. The SUA Utilities and SDK package is downloaded from Microsoft's Web site and installed separately. It provides a UNIX-like environment that includes such programs as **grep**, **ksh**, **ls**, and **ps**.

To install SUA and the SUA Utilities and SDK, follow these steps:

1. Add the Subsystem for UNIX-based Applications feature using the Server Manager (available in **Administrative Tools**). Select **Features** then **Add features**. Check **Subsystem for UNIX-based Applications** in the Add Features Wizard, click **Next**, and **Install**.
2. Download the SUA Utilities and SDK using the shortcut available in the Start menu. Click **Start**, **All Programs**, **Subsystem for UNIX-based Applications**, and then **Download Utilities for Subsystem for UNIX-based Applications**. Follow the download instructions provided.
3. Run the SUA Utilities and SDK installation package. The default settings are not sufficient for GPFS, so you must choose **Custom Installation** when given the option by the Setup Wizard. Include at a minimum the **Base Utilities**, **GNU Utilities**, and **Perl** components (see Figure 8). On the Security Settings panel, check all of the available options: **Enable SuToRoot behavior for SUA programs**, **Enable setuid behavior for SUA programs**, and **Change the default behavior to case sensitive** (see Figure 9 on page 78).

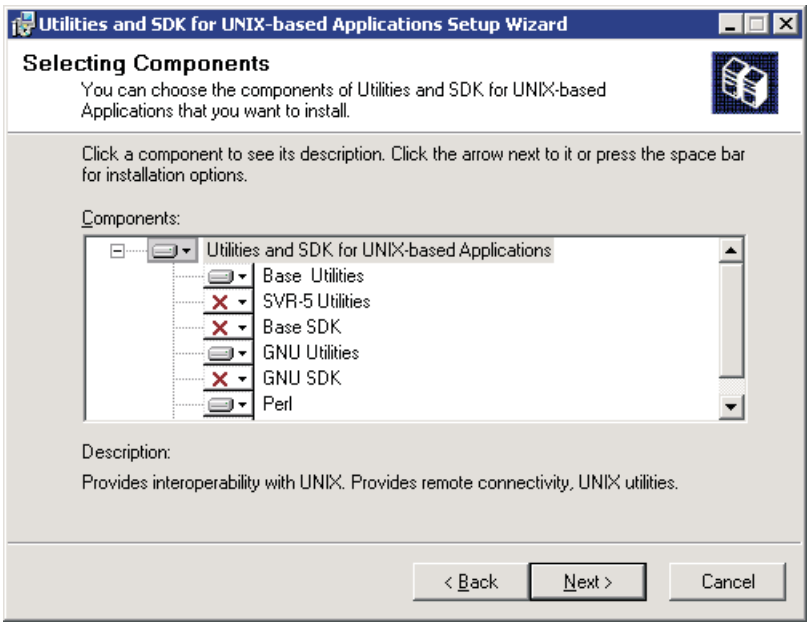

Figure 8. Utilities and SDK for UNIX-based Applications Setup Wizard - Selecting Components

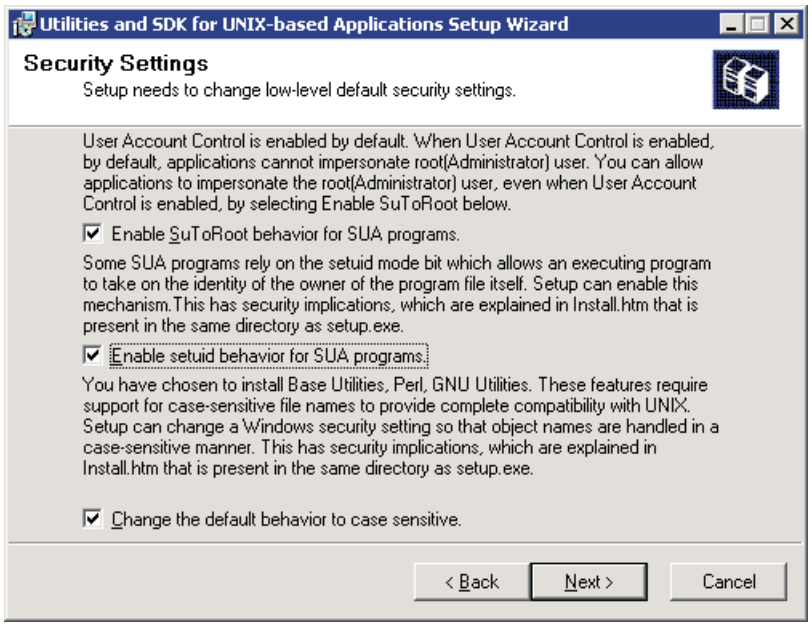

Figure 9. Utilities and SDK for UNIX-based Applications Setup Wizard - Security Settings

4. Complete the installation and reboot if prompted.

Downloading and installing SUA hotfix updates

Microsoft provides hotfix updates that improve the reliability of Subsystem for UNIX-based Applications (SUA) in support of GPFS. Download and install the updates after SUA is installed.

For the latest hotfixes, see the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

You can verify that the updates are installed by opening **Programs and Features** in the Control Panel. Select **View installed updates** in the task list.

Installing the TraceFmt program

GPFS diagnostic tracing (**mmtracectl**) on Windows uses a Microsoft program call **TraceFmt.exe**. This program is not included with Windows but can be downloaded from Microsoft. You only need **TraceFmt.exe** for tracing support; it is not required for normal GPFS operations.

TraceFmt.exe is included with the Windows Driver Kit (WDK) and the Windows SDK (the current version is called the Microsoft Windows SDK for Windows 7 and .NET Framework 4.) You can download either of these packages from the Microsoft Download Center (www.microsoft.com/download).

To allow GPFS diagnostic tracing on Windows using the WDK, follow these steps:

1. Download the Windows Driver Kit (WDK) from Microsoft.
2. Install the Tools feature of the WDK on some system to obtain a copy of **TraceFmt.exe**. The program is in the **tools\tracing\amd64** directory (the full path might be **C:\WinDDK\7600.16385.1\tools\tracing\amd64\tracefmt.exe**).
3. Copy **TraceFmt.exe** to the **%SystemRoot%** directory (for example, **C:\Windows**) or some other directory included in the **PATH** environment variable for all users.

For additional information about GPFS diagnostic tracing, see the topic about the GPFS trace facility in *GPFS: Problem Determination Guide*.

Procedure for installing GPFS on Windows nodes

IBM provides GPFS as a Windows Installer package (MSI), which allows both interactive and unattended installations. Perform the GPFS installation steps as the Administrator or some other member of the Administrators group.

Before installing GPFS on Windows nodes, verify that all the installation prerequisites have been met.

For more information, see "Installing GPFS prerequisites" on page 75.

To install GPFS, follow these steps:

1. Run the installation package from the product media and accept the license. The package name includes the GPFS version and Windows operating system it supports (for example, **gpfs-3.4.0.0-WindowsServer2008-x64.msi**).
2. Download and install the latest service level of GPFS from the GPFS support and downloads site (http://www14.software.ibm.com/webapp/set2/sas/f/gpfs/home.html).

For more information, refer to Chapter 4, "Steps to establishing and starting your GPFS cluster," on page 59.

GPFS update packages might require that you first remove the current GPFS installation from the system. You cannot install an update package without first installing the full installation package from the product media.

The GPFS installation package provides some options that you can select by installing GPFS from a Command Prompt. Property values control the options. The command line syntax is:

```
msiexec.exe /package gpfs-....msi [Optional Parameters] [Property=Value]
```

The following property values are available:

AgreeToLicense=yes
: This option allows GPFS to support an installation procedure that does not require user input. It is recommended that you perform at least one interactive installation on a typical system before attempting an unattended installation. This will help identify any system configuration issues that could cause the installation to fail.

  The following command installs GPFS without prompting you for input or generating any error dialog boxes:

  ```
  msiexec.exe /package gpfs-3.4.0.0-WindowsServer2008-x64.msi /passive AgreeToLicense=yes
  ```

  The **msiexec.exe** executable file supports display options other than **/passive**. See **msiexec** documentation for details.

RemoteShell=no
: This option is equivalent to running **mmwinservctl set –remote-shell no**, but it performs this configuration change before **mmwinserv** initially starts. This option is available to satisfy security policies that restrict network communication protocols.

You can verify that GPFS is installed correctly by opening the **Programs and Features** control panel. "IBM General Parallel File System" should be included in the list of installed programs. The program's version should match the version of the update package.

The GPFS software license agreement is shipped and is viewable electronically. The license agreement will remain available in the **%SystemRoot%\SUA\usr\lpp\mmfs\lap\license** directory for future access.

Running GPFS commands

Once GPFS is installed, the account you use for GPFS administrative operations (such as creating a cluster and file systems) will depend on your cluster type.

For Windows clusters, run GPFS commands as a member of the **Domain Admins** group. For clusters with both Windows and UNIX nodes, run GPFS commands as **root**, a special domain user account described in "Creating the GPFS administrative account" on page 81.

You can run GPFS commands from either a **Windows Command Prompt** or an SUA shell such as **ksh**. Open a new Command Prompt or SUA shell after installing GPFS so that it uses the updated **PATH** environment variable required to execute GPFS commands.

GPFS Admin Command Prompt and **GPFS Admin Korn Shell** under the **IBM General Parallel File System** Start menu group open command line sessions configured to run in Admin Mode. These program links are essentially the same as starting a Command Prompt or Korn Shell using the Run as **Administrator** option. Depending on your system configuration, a GPFS administrative user may issue commands while in Admin Mode in order to have the necessary permissions

Configuring a mixed Windows and UNIX cluster

For GPFS clusters that include both Windows and UNIX nodes, this topic describes the additional configuration steps needed beyond those described in "Installing GPFS prerequisites" on page 75.

For mixed clusters, perform the following steps:

1. Optionally, install and configure identity mapping on your Active Directory domain controller (see "Identity Management for UNIX (IMU)").
2. Create the root administrative account (see "Creating the GPFS administrative account" on page 81).
3. Edit the Domain Group Policy to give root the right to log on as a service (see "Allowing the GPFS administrative account to run as a service" on page 81.
4. Configure the GPFS Administration service (mmwinserv) to run as root (see "Configuring the GPFS Administration service" on page 81).
5. Install and configure OpenSSH (see "Installing and configuring OpenSSH" on page 82).

Complete this process before performing configuration steps common to all GPFS supported platforms.

Identity Management for UNIX (IMU)

GPFS can exploit a Windows Server feature called Identity Management for UNIX (IMU) to provide consistent identities among all nodes in a cluster.

GPFS expects that all Windows nodes in a cluster are members of the same Active Directory domain. This gives domain users a consistent identity and consistent file access rights independent of the system they are using.

In addition, GPFS can exploit the Identity Management for UNIX (IMU) service for mapping users and groups between Windows and UNIX. IMU is an optional component of Microsoft Windows Server (starting with Server 2003 R2) that can be installed on domain controllers. GPFS does not require IMU.

For IMU installation and configuration information, see the topic about identity management on Windows in the *GPFS: Advanced Administration Guide*.

Creating the GPFS administrative account

GPFS uses an administrative account in the Active Directory domain named **root** in order to interoperate with UNIX nodes in the cluster. Create this administrative account as follows:

1. Create a domain user with the logon name **root**.
2. Add user **root** to the **Domain Admins** group.
3. In **root Properties/Profile/Home/LocalPath**, define a **HOME** directory such as **C:\Users\root\home** that does not include spaces in the path name and is not the same as the profile path.
4. Give **root** the right to log on as a service as described in "Allowing the GPFS administrative account to run as a service."

Step 3 is required for the SUA environment (described in "Installing the Subsystem for UNIX-based Applications (SUA)" on page 77) to operate correctly. Avoid using a path that contains a space character in any of the path names. Also avoid using **root**'s profile path (for example, **C:\User\root**). OpenSSH requires specific permissions on this directory, which can interfere with some Windows applications.

You may need to create the **HOME** directory on each node in your GPFS cluster. Make sure that **root** owns this directory.

For more information, see SUA / SFU FAQs (http://www.interopsystems.com/community/faqs.aspx).

Allowing the GPFS administrative account to run as a service

Clusters that depend on a **root** account to interoperate with UNIX nodes in a cluster will need to configure the GPFS Administrative Service (mmwinserv) to run as the **root** account. For this, **root** needs to be assigned the right to log on as a service. See "Configuring the GPFS Administration service" for details.

The right to log on as a service is controlled by the Local Security Policy of each Windows node. You can use the Domain Group Policy to set the Local Security Policy on all Windows nodes in a GPFS cluster.

The following procedure assigns the *log on as a service* right to an account when the domain controller is running on Windows Server 2008:

1. Open **Group Policy Management** (available under **Administrative Tools**).
2. In the console tree, expand ***Forest name*/Domains/*Domain name*/Group Policy Objects**.
3. Right click **Default Domain Policy** and select **Edit**.
4. In the console tree of the Group Policy Management Editor, expand down to **Computer Configuration/Policies/Windows Settings/Security Settings/Local Policies/User Rights Assignment**.
5. Double click the **Log on as a service policy**.
6. Check **Define these policy settings if necessary**.
7. Use **Add User or Group...** to include the ***DomainName*\root** account in the policy, then click **OK**.

Refer to your *Windows Server* documentation for a full explanation of Local Security Policy and Group Policy Management.

Configuring the GPFS Administration service

GPFS for Windows includes a service called **mmwinserv**. In the Windows Services management console, this service has the name GPFS Administration. **mmwinserv** supports GPFS operations such as **autoload** and remote command execution in Windows GPFS clusters. The Linux and AIX versions of GPFS do not have a similar component. The **mmwinserv** service is used on all Windows nodes starting with GPFS 3.3.

The GPFS installation package configures **mmwinserv** to run using the default **LocalSystem** account. This account supports Windows GPFS clusters. For clusters that include both Windows and UNIX nodes, you must configure mmwinserv to run as **root**, the GPFS administrative account. Unlike **LocalSystem**, **root** can access the Identity Management for UNIX (IMU) service and can access other GPFS nodes as required by some cluster configurations.

For information on IMU, see the *Identity management on Windows* topic in the *GPFS: Advanced Administration Guide*. For information on supporting administrative access to GPFS nodes, see the *Requirements for administering a GPFS file system* topic in the *GPFS: Administration and Programming Reference*.

Before configuring mmwinserv to run as **root**, you must first grant **root** the right to run as a service. For details, see "Allowing the GPFS administrative account to run as a service" on page 81.

Use the GPFS command **mmwinservctl** to set and maintain the GPFS Administration service's log on account. **mmwinservctl** must be run on a Windows node. You can run **mmwinservctl** to set the service account before adding Windows nodes to a cluster. You can also use this command to change or update the account on nodes that are already in a cluster. GPFS can be running or stopped when executing **mmwinservctl**, however, refrain from running other GPFS administrative commands at the same time.

In this example, **mmwinservctl** configures three nodes before they are added to a GPFS cluster containing both Windows and UNIX:

```
mmwinservctl set -N node1,node2,node3 --account mydomain/root --password mypwd --remote-shell no
```

Whenever root's password changes, the **mmwinserv** logon information needs to be updated to use the new password. The following command updates on all Windows nodes in a cluster with a new password:

```
mmwinservctl set -N all --password mynewpwd
```

As long as **mmwinserv** is running, the service will not be affected by an expired or changed password and GPFS will continue to function normally. However, GPFS will not start after a system reboot when **mmwinserv** is configured with an invalid password.

For more information about the **mmwinservctl** command, see the *GPFS: Administration and Programming Reference*.

Installing and configuring OpenSSH

For current information about installing OpenSSH, see the GPFS FAQ information at the following URL: http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.pdf

Once OpenSSH is installed, the GPFS administrative account **root**, needs to be configured so that it can issue **ssh** and **scp** commands without requiring a password and without producing any extraneous messages. This kind of passwordless access is required from any node used for GPFS administration to all other nodes in the cluster.

A final step is required for ssh to function correctly in the SUA environment. Run **regpwd** on each Windows node while logged on as **root**. This command records **root**'s password so that remote ssh sessions have full privileges. ssh will work without running **regpwd** (or with a **regpwd** that is set with the wrong password); however, some GPFS commands will fail. If **root**'s password changes, **regpwd** needs to be updated on each Windows node to stay synchronized with the current password.

You can verify that **regpwd** is set correctly by running **mmwinutil verify-com** in an ssh session. For example:

```
ssh winnode1 /usr/lpp/mmfs/bin/mmwinutil verify-com
Okay
```

For additional information, see "Requirements for administering a GPFS file system" in the *GPFS: Administration and Programming Reference* and the "Troubleshooting Windows" topic in the *GPFS: Problem Determination Guide*.

Configuring the Windows HPC server

In order for Windows HPC Server to function with GPFS, disable dynamic hosts file and re-enable dynamic DNS updates by doing the following:

1. On all nodes in the Windows HPC cluster, open the hosts file **%systemroot%\system32\drivers\etc\hosts** and change **ManageFile = true** to **ManageFile = false**.
2. On the HPC head node, execute the following from HPC PowerShell, assuming all nodes are part of the head node's Active Directory domain.

   Go to **Start>All Programs>Microsoft HPC Pack>HPC PowerShell** and execute:

   ```
   Set-HpcNetwork -EnterpriseDnsRegistrationType WithConnectionDnsSuffix
   ```

Chapter 8. Migration, coexistence and compatibility

To migrate to GPFS 3.4, first consider whether you are migrating from GPFS 3.3 or from an earlier release of GPFS, and then consider coexistence and compatibility issues.

GPFS supports a limited form of backward compatibility between two adjacent GPFS releases. Limited backward compatibility allows you to temporarily operate with a mixture of GPFS 3.4 and GPFS 3.3 nodes:

- Within a cluster this enables you to perform a rolling upgrade to the new GPFS 3.4 version of the code provided your current version is GPFS 3.3.
- In a multicluster environment this allows the individual clusters to be upgraded on their own schedules. Access to the file system data can be preserved even though some of the clusters may still be running GPFS 3.3 level.

Rolling upgrades allow you to install new GPFS code one node at a time without shutting down GPFS on other nodes. However, you must upgrade all nodes within a short time. The time dependency exists because some GPFS 3.4 features become available on each node as soon as the node is upgraded, while other features will not become available until you upgrade all participating nodes. Once all nodes have been migrated to the new code, you must finalize the migration by running the commands **mmchconfig release=LATEST** and **mmchfs -V full** (or **mmchfs -V compat**). Also, certain new features may require the use of **mmmigratefs** on the unmounted file system to enable them.

For the latest information on migration, coexistence, and compatibility, see the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

GPFS migration consists of these topics:

Migrating to GPFS 3.4 from GPFS 3.3

GPFS 3.4 supports node-at-a-time migration if the previous nodes in the cluster are running GPFS 3.3. GPFS 3.3 nodes can coexist and interoperate with nodes that are running GPFS 3.4. However, some new functions that depend on format changes will not be available until all nodes have been migrated.

To migrate a cluster to GPFS 3.4 from GPFS 3.3, perform these steps:

1. Stop all user activity in the file systems on the designated node.

   Note: For information about how to stop and then unmount NFS running over GPFS file systems, see the topic about unmounting a file system after NFS export in *GPFS: Administration and Programming Reference*.
2. Cleanly unmount the mounted GPFS file system. Do not use force unmount on the designated node.

3. Follow any local administrative backup procedures to ensure protection of your file system data in the event of a failure.
4. Stop GPFS on the node to be migrated in the cluster, for example:
   ```
   mmshutdown -N k164n04
   ```
5. Copy the installation images and install the GPFS licensed program as described in Chapter 5, "Installing GPFS on Linux nodes," on page 61, Chapter 6, "Installing GPFS on AIX nodes," on page 67, or Chapter 7, "Installing GPFS on Windows nodes," on page 71.
6. Start GPFS on the designated node in the cluster, for example:
   ```
   mmstartup -N k164n04
   ```
7. Mount the file systems if this is not done automatically when the GPFS daemon starts.

When all nodes in the cluster have been successfully migrated to the new GPFS level, proceed to "Completing the migration to a new level of GPFS."

Migrating to GPFS 3.4 from GPFS 3.2, GPFS 3.1, or GPFS 2.3

Node-at-a-time migration is not available when migrating to GPFS 3.4 from GPFS 3.2, GPFS 3.1, or GPFS 2.3. The cluster must be completely shut down and *all* nodes migrated at the same time. If this is not acceptable, and your current level is GPFS 3.2, you may want to consider an intermediate migration to GPFS 3.3 first.

To migrate a cluster to GPFS 3.4 from GPFS 3.2, GPFS 3.1 or GPFS 2.3, perform these steps:

1. Stop all user activity in the file systems.

   Note: For information about how to stop and then unmount NFS running over GPFS files systems, see the topic about unmounting a file system after NFS export in *GPFS: Administration and Programming Reference*.
2. Cleanly unmount the mounted GPFS file system. Do not use force unmount.
3. Follow any local administrative backup procedures to ensure protection of your file system data in the event of a failure.
4. Stop GPFS on all nodes in the cluster:
   ```
   mmshutdown -a
   ```
5. Copy the installation images and install the GPFS product on each node in the cluster as described in Chapter 5, "Installing GPFS on Linux nodes," on page 61 or Chapter 6, "Installing GPFS on AIX nodes," on page 67.
6. Start GPFS on all nodes in the cluster:
   ```
   mmstartup -a
   ```
7. Mount the file systems if this is not done automatically when the GPFS daemon starts.
8. Proceed to "Completing the migration to a new level of GPFS."

Completing the migration to a new level of GPFS

You should operate with the new level of the GPFS licensed program until you are sure that you want to permanently migrate. If you decide not to migrate, you can revert to the previous level of GPFS, see "Reverting to the previous level of GPFS" on page 87 for more information.

Once the new level of GPFS is satisfactory for your environment, you must complete migration of both the cluster configuration data and all file systems.

After you have migrated *all* nodes to the latest GPFS licensed program:

1. Migrate the cluster configuration data and enable new cluster-wide functionality:
   ```
   mmchconfig release=LATEST
   ```

The **mmchconfig** command will list the names of the nodes that are not available or cannot be reached. If this is the case, correct the problem and reissue the command until all nodes can be verified and the command completes successfully.

2. If you have not already done so, assign an appropriate GPFS license to each of the nodes in the cluster. See "GPFS license designation" on page 38 for a detailed description of the GPFS license types.

   To see what the minimum required GPFS license is for each of the nodes in the cluster, issue:

   ```
   mmlslicense -L
   ```

   To assign a GPFS server license to the nodes that require it, issue:

   ```
   mmchlicense server -N nodeList
   ```

   To assign a GPFS client license to the nodes that require it, issue:

   ```
   mmchlicense client -N nodeList
   ```

3. Enable backward-compatible format changes or migrate all file systems to the latest metadata format changes.

   Attention: Before continuing with this step, it is important to understand the differences between **mmchfs -V compat** and **mmchfs -V full**:

   - If you issue **mmchfs -V compat**, only changes that are backward compatible with GPFS 3.3 will be enabled. Nodes in remote clusters that are running GPFS 3.3 will still be able to mount the file system. Nodes running GPFS 3.2 or earlier will no longer be able to mount the file system.
   - If you issue **mmchfs -V full**, all new functions that require different on-disk data structures will be enabled. Nodes in remote clusters running an older GPFS version will no longer be able to mount the file system. If there are any nodes running an older GPFS version that have the file system mounted at the time this command is issued, the **mmchfs** command will fail.

   Certain new file system features may require additional processing that cannot be handled by the **mmchfs -V** command alone. To fully activate such features, in addition to **mmchfs -V**, you will also have to run the **mmmigratefs** command. An example of such a feature in GPFS 3.4 is the support for fast extended attributes.

   To enable backward-compatible format changes, issue the following command:

   mmchfs *filesystem* **-V compat**

   To migrate all file systems to the latest metadata format changes, issue the following command:
   mmchfs *filesystem* **-V full**

   To activate fast extended attributes, unmount the file system and issue the following command:
   mmmigratefs *filesystem* **--fastea**

4. Decide whether you want to change the value of the **adminMode** configuration parameter. For a detailed description, see the section about the requirements for administering a GPFS file system in the *GPFS: Administration and Programming Reference*.

Reverting to the previous level of GPFS

If you should decide not to continue the migration to the latest level of GPFS, and you have not yet issued the **mmchfs -V** command, you can reinstall the earlier level of GPFS.

Important: Once a file system has been migrated explicitly by issuing the **mmchfs -V full** command, the disk images can no longer be read by a prior version of GPFS. You will be required to re-create the file system from the backup media and restore the content if you choose to go back after this command has been issued. The same rules apply for file systems that are newly created with GPFS 3.4.

You can revert back to GPFS 3.3 or GPFS 3.2. Earlier GPFS releases are no longer supported by IBM.

The procedure differs depending on whether you have issued the **mmchconfig release=LATEST** command or not.

Reverting to a previous level of GPFS when you have *not* issued mmchconfig release=LATEST

If you have **not** issued the **mmchconfig release=LATEST** command, perform these steps:

1. Stop all user activity in the file systems.
2. Cleanly unmount all mounted GPFS file systems. Do not use force unmount.
3. Stop GPFS on all nodes in the cluster:
   ```
   mmshutdown -a
   ```
4. Run the appropriate de-installation program to remove GPFS from each node in the cluster. For example:
   - For Linux nodes:
     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs
     ```
   - For AIX nodes:
     ```
     installp -u gpfs
     ```
   - For Windows nodes, open the **Programs and Features** control panel and remove **IBM General Parallel File System**.

   For the remaining steps, see the appropriate for your release *GPFS: Concepts, Planning, and Installation Guide* at publib.boulder.ibm.com/infocenter/clresctr/topic/ com.ibm.cluster.gpfs.doc/ gpfsbooks.html.
5. Copy the installation images of the previous GPFS licensed program on all affected nodes.
6. Install the original install images and all required PTFs.
7. For Linux nodes running GPFS, you must rebuild the GPFS portability layer.
8. Reboot all nodes.

Reverting to a previous level of GPFS when you *have* issued mmchconfig release=LATEST

If you *have* issued the **mmchconfig release=LATEST** command, you must rebuild the cluster. Perform these steps:

1. Stop all user activity in the file systems.
2. Cleanly unmount all mounted GPFS file systems. Do not use force unmount.
3. Stop GPFS on all nodes in the cluster:
   ```
   mmshutdown -a
   ```
4. Export the GPFS file systems by issuing the **mmexportfs** command:
   ```
   mmexportfs all -o exportDataFile
   ```
5. Delete the cluster:
   ```
   mmdelnode -a
   ```
6. Run the appropriate de-installation program to remove GPFS from each node in the cluster. For example:
   - For Linux nodes:
     ```
     rpm -e gpfs.gpl gpfs.base gpfs.docs
     ```
   - For AIX nodes:
     ```
     installp -u gpfs
     ```
   - For Windows nodes, open the **Programs and Features** control panel and remove **IBM General Parallel File System**.

   For the remaining steps, see the appropriate for your release GPFS: *Concepts, Planning, and Installation Guide* at publib.boulder.ibm.com/infocenter/clresctr/topic/ com.ibm.cluster.gpfs.doc/ gpfsbooks.html.
7. Copy the installation images of the previous GPFS licensed program on all affected nodes.

8. Install the original installation images and all required PTFs.
9. For Linux nodes running GPFS, you must rebuild the GPFS portability layer.
10. Reboot all nodes.
11. Recreate your original cluster using the **mmcrcluster** command.
12. Use **mmchconfig** to restore any of the configuration settings that were previously set.
13. Import the file system information using the **mmimportfs** command. Specify the file created by the **mmexportfs** command from Step 4 on page 88 above:

    ```
    mmimportfs all -i exportDataFile
    ```

14. Start GPFS on all nodes in the cluster, issue:

    ```
    mmstartup -a
    ```

15. Mount the file systems if this is not done automatically when the GPFS daemon starts.

Coexistence considerations

Each GPFS cluster can have multiple GPFS file systems that coexist on the cluster, but function independently of each other. In addition, each file system might have different data management programs.

Note: The GPFS Data Management API (DMAPI) and GPFS file system snapshots can coexist; however, access to the files in a snapshot using DMAPI is restricted. See the *General Parallel File System: Data Management API Guide* for more information.

Compatibility considerations

All applications that ran on the previous release of GPFS will run on the new level of GPFS. File systems that were created under the previous release of GPFS can be used under the new level of GPFS.

Considerations for IBM Tivoli Storage Manager for Space Management

Migrating to GPFS 3.3 or beyond requires consideration for IBM Tivoli Storage Manager for Space Management. IBM Tivoli Storage Manager for Space Management requires that all nodes in a cluster are configured with a DMAPI file handle size of 32 bytes.

During migration, it is possible to have older versions of GPFS that have a DMAPI file handle size of 16 bytes. Until all nodes in the cluster have been updated to the latest release and the DMAPI file handle size has been changed to 32 bytes, IBM Tivoli Storage Manager for Space Management must be disabled. Run **mmlsconfig dmapiFileHandleSize** to see what value is set.

After all nodes in the cluster are upgraded to the latest release and you change the DMAPI file handle size to 32 bytes, you can enable IBM Tivoli Storage Manager for Space Management.

The DMAPI file handle size is configured with the **dmapiFileHandleSize** option. For more information about this option, see the topic "GPFS configuration options for DMAPI" in the *General Parallel File System: Data Management API Guide*.

Applying maintenance to your GPFS system

Before applying maintenance to your GPFS system, there are several things you should consider.

Remember that:

1. There is limited interoperability between GPFS 3.4 nodes and GPFS 3.3 nodes. This function is intended for short-term use. You will not get the full functions of GPFS 3.4 until all nodes are using GPFS 3.4.

2. Interoperability between maintenance levels will be specified on a per-maintenance-level basis. See the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html) for the latest on service information for GPFS.
3. Maintenance images for GPFS Linux retrieved from the web are named differently from the installation images of the GPFS Linux product. Maintenance images contain the word **update** in their name, for example: **gpfs.base-3.3-1-1.i386.update.rpm**.
4. When applying maintenance, the GPFS file systems will need to be unmounted and GPFS shut down on the node being upgraded. Any applications using a GPFS file system should be shut down before applying maintenance to avoid application errors or possible system halts.

   Note: If the AIX Alternate Disk Install process is used, the new image is placed on a different disk. The running install image is not changed, and GPFS is not shut down. The new image is not used until the node is rebooted.
5. For the latest service information, see the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

To download fixes, go to General Parallel File System (GPFS) Support and downloads for AIX, Linux, and Windows on IBM Systems (https://www14.software.ibm.com/webapp/set2/sas/f/gpfs/home.html) and obtain the fixes for your hardware and operating system.

To install the latest fixes:

- For Linux nodes, see Chapter 5, “Installing GPFS on Linux nodes,” on page 61.
- For AIX nodes, see Chapter 6, “Installing GPFS on AIX nodes,” on page 67.
- For Windows nodes, see Chapter 7, “Installing GPFS on Windows nodes,” on page 71.

Chapter 9. Configuring and tuning your system for GPFS

In addition to configuring your GPFS cluster, you need to configure and tune your system.

Note: See "GPFS cluster creation considerations" on page 34 for more information.

Values suggested here reflect evaluations made at the time this documentation was written. For the latest system configuration settings, see the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

Additional GPFS and system configuration and tuning considerations include:

1. "General system configuration and tuning considerations"
2. "Linux configuration and tuning considerations" on page 94
3. "AIX configuration and tuning considerations" on page 96
4. "Configuring Windows" on page 76

See the *General Parallel File System: Advanced Administration Guide* for information on:

- The **mmpmon** command for analyzing I/O performance on a per-node basis in *Monitoring GPFS I/O performance with the **mmpmon** command*
- Using multiple token servers.

General system configuration and tuning considerations

You need to take into account some general system configuration and tuning considerations. This topic points you to the detailed information.

For the latest system configuration settings, see the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

Configuration and tuning considerations for all systems include:

1. "Clock synchronization"
2. "GPFS administration security"
3. "Cache usage" on page 92
4. "GPFS I/O" on page 93
5. "Access patterns" on page 94
6. "Aggregate network interfaces" on page 94
7. "Swap space" on page 94

Clock synchronization

The clocks of all nodes in the GPFS cluster must be synchronized. If this is not done, NFS access to the data, as well as other GPFS file system operations may be disrupted.

GPFS administration security

Before administering your GPFS file system, make certain that your system has been properly configured for security. This includes:

- Assigning root authority to perform all GPFS administration tasks except:
  - Tasks with functions limited to listing GPFS operating characteristics.

  - Tasks related to modifying individual user file attributes.
- Establishing the authentication method between nodes in the GPFS cluster.
  - Until you set the authentication method, you cannot issue any GPFS commands.
- Designating a remote communication program for remote shell and remote file copy commands.
  - The default remote communication commands are **rcp** and **rsh**. You can designate any other remote commands if they have the same syntax.
  - Regardless of which remote commands have been selected, the nodes that you plan to use for administering GPFS must be able to execute commands on any other node in the cluster without the use of a password and without producing any extraneous messages.

Cache usage

GPFS creates a number of cache segments on each node in the cluster. The amount of cache is controlled by three attributes. These attributes have default values at cluster creation time and may be changed through the **mmchconfig** command:

pagepool

The GPFS pagepool is used to cache user data and file system metadata. The pagepool mechanism allows GPFS to implement read as well as write requests asynchronously. Increasing the size of pagepool increases the amount of data or metadata that GPFS can cache without requiring synchronous I/O. The amount of memory available for GPFS pagepool on a particular node may be restricted by the operating system and other software running on the node.

The optimal size of the pagepool depends on the needs of the application and effective caching of its re-accessed data. For systems where applications access large files, reuse data, benefit from GPFS prefetching of data, or have a random I/O pattern, increasing the value for **pagepool** may prove beneficial. However, if the value is set too large, GPFS will not start. See the *GPFS: Problem Determination Guide* and search on *GPFS daemon will not come up*.

To change the pagepool to 4GB:

```
mmchconfig pagepool=4G
```

maxFilesToCache

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

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

maxFilesToCache × **3 KB**

Valid values of **maxFilesToCache** range from 1 to 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:

maxStatCache × **400 bytes**

Valid values of **maxStatCache** range from 0 to 10,000,000. 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 × 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** × 3 KB), and the memory for the stat cache (**maxStatCache** × 400 bytes) together. The combined amount of memory to hold inodes, control data structures, and the stat cache is limited to 50% of the physical memory on a node running GPFS.

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

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

For further information on these cache settings for GPFS, refer to "GPFS and memory" on page 14.

The GPFS token system's affect on cache settings

Lock tokens play a role in maintaining cache consistency between nodes. A token allows a node to cache data it has read from disk, because the data cannot be modified elsewhere without revoking the token first. Each token manager can handle approximately 300,000 different file tokens (this number depends on how many distinct byte-range tokens are used when multiple nodes access the same file). If you divide the 300,000 by the number of nodes in the GPFS cluster you get a value that should approximately equal **maxFilesToCache** (the total number of different files that can be cached at one time) + *maxStatCache* (additional pageable memory to cache file attributes that are not currently in the regular file cache).

The configuration parameter:

- **maxFilesToCache** should be large enough to handle the number of concurrently open files plus allow caching of recently used files.
- **maxStatCache** defaults to **4 x maxFilesToCache** but can be set independently to balance the speed of **ls -l** calls with the memory load on the token manager memory.
- **maxStatCache** can be set higher on user-interactive-nodes and smaller on dedicated compute-nodes, since **ls -l** performance is mostly a human response issue.
- **maxFilesToCache** and **maxStatCache** are indirectly affected by the **distributedTokenServer** configuration parameter because distributing the tokens across multiple token servers might allow keeping more tokens than if a file system has only one token server.

GPFS I/O

The **maxMBpS** option determines the maximum amount of I/O in MB that can be submitted by GPFS per second. If the default value is not adjusted accordingly it will affect GPFS performance. Note that setting this number too high can have an adverse effect on performance of the system since overrunning the capabilities of the I/O bus or network adapter tends to drastically degrade throughput. This number is normally set after empirical study to determine your nodes I/O bandwidth limits. The default value is 150 MB per second. To change maxMBpS to 500 MB per second:

```
mmchconfig maxMBpS=500
```

Access patterns

GPFS attempts to recognize the pattern of accesses (such as strided sequential access) that an application makes to an open file. If GPFS recognizes the access pattern, it will optimize its own behavior. For example, GPFS can recognize sequential reads and will retrieve file blocks before they are required by the application. However, in some cases GPFS does not recognize the access pattern of the application or cannot optimize its data transfers. In these situations, you may improve GPFS performance if the application explicitly discloses aspects of its access pattern to GPFS through the **gpfs_fcntl()** library call.

Aggregate network interfaces

It is possible to aggregate multiple physical Ethernet interfaces into a single virtual interface. This is known as *Channel Bonding* on Linux and *EtherChannel/IEEE 802.3ad Link Aggregation* on AIX. GPFS supports using such aggregate interfaces. The main benefit is increased bandwidth. The aggregated interface has the network bandwidth close to the total bandwidth of all its physical adapters. Another benefit is improved fault tolerance. If a physical adapter fails, the packets are automatically sent on the next available adapter without service disruption.

EtherChannel and IEEE802.3ad each requires support within the Ethernet switch. Refer to the product documentation for your switch to determine if EtherChannel is supported.

For details on how to configure EtherChannel and IEEE 802.3ad Link Aggregation:

1. Go to
   `publib16.boulder.ibm.com/pseries/en_US/aixbman/commadmn/ tcp_etherchannel.htm#yu528frokferg`
2. Search on *Configuring EtherChannel*

Hint: Make certain that the switch ports are configured for **LACP** (the default is **PAGP**).

For details on how to verify whether the adapter and the switch are operating with the correct protocols for IEEE 802.3ad:

1. Go to
   `publib16.boulder.ibm.com/pseries/en_US/aixbman/commadmn/ tcp_etherchannel.htm#yu528frokferg`
2. Search on *Troubleshooting IEEE 802.3ad*.

For additional service updates regarding the use of EtherChannel:

1. Go to www.ibm.com/support
2. In the **Search technical support** box, enter the search term *EtherChannel*
3. Click **Search**

Hint: A useful command for troubleshooting, where device is the Link Aggregation device, is:

```
entstat -d device
```

Swap space

It is highly suggested that a sufficiently large amount of swap space is configured. While the actual configuration decisions should be made taking into account the memory requirements of other applications, it is suggested to configure at least as much swap space as there is physical memory on a given node.

Linux configuration and tuning considerations

Configuration and tuning considerations for the Linux nodes in your system include the use of the **updatedb** utility, the **vm.min_free_kbytes** kernel tunable, and several other options that can improve GPFS performance.

For the latest system configuration settings, see the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html

For more configuration and tuning considerations for Linux nodes, see the following topics:

1. "updatedb considerations"
2. "Memory considerations"
3. "GPFS helper threads"
4. "Communications I/O"
5. "Disk I/O" on page 96

updatedb considerations

On some Linux distributions, the system is configured by default to run the file system indexing utility **updatedb** through the **cron** daemon on a periodic basis (usually daily). This utility traverses the file hierarchy and generates a rather extensive amount of I/O load. For this reason, it is configured by default to skip certain file system types and nonessential file systems. However, the default configuration does not prevent **updatedb** from traversing GPFS file systems. In a cluster this results in multiple instances of **updatedb** traversing the same GPFS file system simultaneously. This causes general file system activity and lock contention in proportion to the number of nodes in the cluster. On smaller clusters, this may result in a relatively short-lived spike of activity, while on larger clusters, depending on the overall system throughput capability, the period of heavy load may last longer. Usually the file system manager node will be the busiest, and GPFS would appear sluggish on all nodes. Re-configuring the system to either make **updatedb** skip all GPFS file systems or only index GPFS files on one node in the cluster is necessary to avoid this problem.

Memory considerations

It is recommended that you adjust the **vm.min_free_kbytes** kernel tunable. This tunable controls the amount of free memory that Linux kernel keeps available (i.e. not used in any kernel caches). When **vm.min_free_kbytes** is set to its default value, on some configurations it is possible to encounter memory exhaustion symptoms when free memory should in fact be available. Setting **vm.min_free_kbytes** to a higher value (Linux **sysctl** utility could be used for this purpose), on the order of magnitude of 5-6% of the total amount of physical memory, should help to avoid such a situation.

See the GPFS Redbooks® papers for more information:

- *GPFS Sequential Input/Output Performance on IBM pSeries® 690* (http://www.redbooks.ibm.com/redpapers/pdfs/redp3945.pdf)
- *Native GPFS Benchmarks in an Integrated p690/AIX and x335/Linux Environment* (http://www.redbooks.ibm.com/redpapers/pdfs/redp3962.pdf)

GPFS helper threads

GPFS uses helper threads, such as prefetchThreads, worker1Threads to improve performance. Since systems vary, it is suggested you simulate an expected workload in GPFS and examine available performance indicators on your system. For instance some SCSI drivers publish statistics in the **/proc/scsi** directory. If your disk driver statistics indicate that there are many *queued requests* it may mean you should throttle back the helper threads in GPFS. Suggested starting points are:

```
mmchconfig prefetchThreads=48
mmchconfig worker1Threads=72
```

Communications I/O

To optimize the performance of GPFS and your network, it is suggested you do the following:

- Enable Jumbo Frames if your switch supports it.

  If GPFS is configured to operate over Gigabit Ethernet, set the MTU size for the communication adapter to 9000.

- Verify **/proc/sys/net/ipv4/tcp_window_scaling** is enabled. It should be by default.
- Tune the TCP window settings by adding these lines to the **/etc/sysctl.conf** file:

```
# increase Linux TCP buffer limits
net.core.rmem_max = 8388608
net.core.wmem_max = 8388608
# increase default and maximum Linux TCP buffer sizes
net.ipv4.tcp_rmem = 4096 262144 8388608
net.ipv4.tcp_wmem = 4096 262144 8388608
# increase max backlog to avoid dropped packets
net.core.netdev_max_backlog=2500
```

After these changes are made to the **/etc/sysctl.conf** file, apply the changes to your system:

1. Issue the **sysctl -p /etc/sysctl.conf** command to set the kernel settings.
2. Issue the **mmstartup -a** command to restart GPFS

Disk I/O

To optimize disk I/O performance, you should consider theses options for NSD servers or other GPFS nodes that are directly attached to a SAN over a Fibre Channel network:

1. The storage server cache settings can impact GPFS performance if not set correctly.
2. When the storage server disks are configured for RAID5, some configuration settings can affect GPFS performance. These settings include:
   - GPFS block size
   - Maximum I/O size of host Fibre Channel (FC) host bus adapter (HBA) device driver
   - Storage server RAID5 stripe size

   Note: For optimal performance, GPFS block size should be a multiple of the maximum I/O size of the FC HBA device driver. In addition, the maximum I/O size of the FC HBA device driver should be a multiple of the RAID5 stripe size.
3. These suggestions may avoid the performance penalty of read-modify-write at the storage server for GPFS writes. Examples of the suggested settings are:
   - 8+P RAID5
     - GPFS block size = 512K
     - Storage Server RAID5 segment size = 64K (RAID5 stripe size=512K)
     - Maximum IO size of FC HBA device driver = 512K
   - 4+P RAID5
     - GPFS block size = 256K
     - Storage Server RAID5 segment size = 64K (RAID5 stripe size = 256K)
     - Maximum IO size of FC HBA device driver = 256K

   For the example settings using 8+P and 4+P RAID5, the RAID5 parity can be calculated from the data written and will avoid reading from disk to calculate the RAID5 parity. The maximum IO size of the FC HBA device driver can be verified using **iostat** or the Storage Server performance monitor. In some cases, the device driver may need to be patched to increase the default maximum IO size.
4. The GPFS parameter **maxMBpS** can limit the maximum throughput of an NSD server or a single GPFS node that is directly attached to the SAN with a FC HBA. Increase the **maxMBpS** from the default value of 150 to 200 (200 MB/s). The **maxMBpS** parameter is changed by issuing the **mmchconfig** command. After this change is made, restart GPFS on the nodes and test both read and write performance of both a single node in addition to a large number of nodes.

AIX configuration and tuning considerations

For the latest system configuration settings, see the GPFS FAQ (http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html).

GPFS use with Oracle

When utilizing GPFS with Oracle, configuration and tuning considerations include:

- When setting up your LUNs, it is important to create the NSD such that they map one to one with a LUN that is a single RAID device.
- For file systems holding large Oracle databases, set the GPFS file system block size through the **mmcrfs** command using the **-B** option, to a large value:
  - 512 KB is generally suggested.
  - 256 KB is suggested if there is activity other than Oracle using the file system and many small files exist which are not in the database.
  - 1 MB is suggested for file systems 100 TB or larger.

  The large block size makes the allocation of space for the databases manageable and has no affect on performance when Oracle is using the Asynchronous I/O (AIO) and Direct I/O (DIO) features of AIX.
- Set the GPFS worker threads through the **mmchconfig prefetchThreads** command to allow the maximum parallelism of the Oracle AIO threads:
  - This setting can be as large as 548.

    The GPFS prefetch threads must be adjusted accordingly through the **mmchconfig prefetchThreads** command as the sum of those two classes of threads must be 550 or less.
  - When requiring GPFS sequential I/O, set the prefetch threads between 50 and 100 (the default is 64), and set the worker threads to have the remainder.

  Note: These changes through the **mmchconfig** command take effect upon restart of the GPFS daemon.
- The number of AIX AIO *kprocs* to create should be approximately the same as the GPFS *worker1Threads* setting.
- The AIX AIO *maxservers* setting is the number of *kprocs* PER CPU. It is suggested to set is slightly larger than *worker1Threads* divided by the number of CPUs. For example if *worker1Threads* is set to 500 on a 32-way SMP, set *maxservers* to 20.
- Set the Oracle database block size equal to the LUN segment size or a multiple of the LUN pdisk segment size.
- Set the Oracle read-ahead value to prefetch one or two full GPFS blocks. For example, if your GPFS block size is 512 KB, set the Oracle blocks to either 32 or 64 16 KB blocks.
- Do not use the **dio** option on the **mount** command as this forces DIO when accessing *all* files. Oracle automatically uses DIO to open database files on GPFS.
- When running Oracle RAC 10g, it is suggested you increase the value for **OPROCD_DEFAULT_MARGIN** to at least 500 to avoid possible random reboots of nodes.

  In the control script for the Oracle CSS daemon, located in **/etc/init.cssd** the value for **OPROCD_DEFAULT_MARGIN** is set to 500 (milliseconds) on all UNIX derivatives except for AIX. For AIX this value is set to 100. From a GPFS perspective, even 500 milliseconds maybe too low in situations where node failover may take up to a minute or two to resolve. However, if during node failure the surviving node is already doing direct IO to the **oprocd** control file, it should have the necessary tokens and indirect block cached and should therefore not have to wait during failover.

Chapter 10. Steps to permanently uninstall 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 on any of the nodes in your cluster and you want to remove all traces of GPFS:

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

1. Unmount all GPFS file systems on all nodes by issuing the **mmumount all -a** command.
2. Issue the **mmdelfs** command for each file system in the cluster to remove GPFS file systems.
3. Issue the **mmdelnsd** command for each NSD in the cluster to remove the NSD volume ID written on sector 2.

   If the NSD volume ID is not removed and the disk is again used with GPFS at a later time, you will receive an error message when issuing the **mmcrnsd** command. See *NSD creation fails with a message referring to an existing NSD* in the *GPFS: Problem Determination Guide*.
4. Issue the **mmshutdown -a** command to shutdown GPFS on all nodes.
5. Uninstall GPFS from each node:
   - For your Linux nodes, run the de-install to remove GPFS for the correct version of the RPM for your hardware platform and Linux distribution. For example:

     ```
     rpm -e gpfs.gpl
     rpm -e gpfs.msg.en_us
     rpm -e gpfs.base
     rpm -e gpfs.docs
     ```
   - For your AIX nodes:

     installp -u gpfs
   - For your Windows nodes, follow these steps:
     a. Open **Programs and Features** in the Control Panel.
     b. Uninstall **IBM General Parallel File System**.
     c. Reboot the system.
     d. From a Command Prompt, run the following command:

        sc.exe delete mmwinserv
6. Remove the **/var/mmfs** and **/usr/lpp/mmfs** directories.
7. Remove all files that start with **mm** from the **/var/adm/ras** directory.
8. Remove **/tmp/mmfs** directory and its content, if present.

Chapter 11. Considerations for GPFS applications

Application design should take into consideration the exceptions to Open Group technical standards with regard to the **stat()** system call and NFS V4 ACLs. Also, a technique to determine whether a file system is controlled by GPFS has been provided.

For more information, see the following topics in the *GPFS: Administration and Programming Reference*:

- Exceptions to Open Group technical standards
- Determining if a file system is controlled by GPFS
- GPFS exceptions and limitation to NFS V4 ACLs

Accessibility features for GPFS

Accessibility features help users who have a disability, such as restricted mobility or limited vision, to use information technology products successfully.

Accessibility features

The following list includes the major accessibility features in GPFS:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

The **IBM Cluster Information Center**, and its related publications, are accessibility-enabled. The accessibility features of the information center are described at Accessibility

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility:

| http://www.ibm.com/able/

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 10504-1785
USA

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

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 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
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment 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 the application programming interfaces 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.

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

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Intel® and Pentium® are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

| Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States,
| other countries, or both.

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

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

Glossary

This glossary defines technical terms and abbreviations used in GPFS documentation. If you do not find the term you are looking for, refer to the index of the appropriate book or view the IBM Glossary of Computing Terms, located on the Internet at: http://www-306.ibm.com/software/globalization/terminology/index.jsp.

B

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

C

cluster
A loosely-coupled collection of independent systems (nodes) organized into a network for the purpose of sharing resources and communicating with each other. See also *GPFS cluster*.

cluster configuration data
The configuration data that is stored on the cluster configuration servers.

cluster manager
The node that monitors node status using disk leases, detects failures, drives recovery, and selects file system managers. The cluster manager is the node with the lowest node number among the quorum nodes that are operating at a particular time.

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

D

Data Management Application Program Interface (DMAPI)
The interface defined by the Open Group's XDSM standard as described in the publication *System Management: Data Storage Management (XDSM) API Common Application Environment (CAE) Specification C429*, The Open Group ISBN 1-85912-190-X.

deadman switch timer
A kernel timer that works on a node that has lost its disk lease and has outstanding I/O requests. This timer ensures that the node cannot complete the outstanding I/O requests (which would risk causing file system corruption), by causing a panic in the kernel.

disk descriptor
A definition of the type of data that the disk contains and the failure group to which this disk belongs. See also *failure group*.

disposition
The session to which a data management event is delivered. An individual disposition is set for each type of event from each file system.

disk leasing
A method for controlling access to storage devices from multiple host systems. Any host that wants to access a storage device configured to use disk leasing registers for a lease; in the event of a perceived failure, a host system can deny access, preventing I/O operations with the storage device until the preempted system has reregistered.

domain
A logical grouping of resources in a network for the purpose of common management and administration.

F

failback
Cluster recovery from failover following repair. See also *failover*.

failover
(1) The assumption of file system duties by another node when a node fails. (2) The process of transferring all control of the ESS to a single cluster in the ESS when the other clusters in the ESS fails. See also *cluster*. (3) The routing of all

transactions to a second controller when the first controller fails. See also *cluster*.

failure group
A collection of disks that share common access paths or adapter connection, and could all become unavailable through a single hardware failure.

fileset A hierarchical grouping of files managed as a unit for balancing workload across a cluster.

file-management policy
A set of rules defined in a policy file that GPFS uses to manage file migration and file deletion. See also *policy*.

file-placement policy
A set of rules defined in a policy file that GPFS uses to manage the initial placement of a newly created file. See also *policy*.

file system descriptor
A data structure containing key information about a file system. This information includes the disks assigned to the file system (*stripe group*), the current state of the file system, and pointers to key files such as quota files and log files.

file system descriptor quorum
The number of disks needed in order to write the file system descriptor correctly.

file system manager
The provider of services for all the nodes using a single file system. A file system manager processes changes to the state or description of the file system, controls the regions of disks that are allocated to each node, and controls token management and quota management.

fragment
The space allocated for an amount of data too small to require a full block. A fragment consists of one or more subblocks.

G

GPFS cluster
A cluster of nodes defined as being available for use by GPFS file systems.

GPFS portability layer
The interface module that each installation must build for its specific hardware platform and Linux distribution.

GPFS recovery log
A file that contains a record of metadata activity, and exists for each node of a cluster. In the event of a node failure, the recovery log for the failed node is replayed, restoring the file system to a consistent state and allowing other nodes to continue working.

I

ill-placed file
A file assigned to one storage pool, but having some or all of its data in a different storage pool.

ill-replicated file
A file with contents that are not correctly replicated according to the desired setting for that file. This situation occurs in the interval between a change in the file's replication settings or suspending one of its disks, and the restripe of the file.

indirect block
A block containing pointers to other blocks.

inode The internal structure that describes the individual files in the file system. There is one inode for each file.

J

journaled file system (JFS)
A technology designed for high-throughput server environments, which are important for running intranet and other high-performance e-business file servers.

junction
A special directory entry that connects a name in a directory of one fileset to the root directory of another fileset.

K

kernel The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.

M

metadata
A data structures that contain access information about file data. These include: inodes, indirect blocks, and directories. These data structures are not accessible to user applications.

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

mirroring
The process of writing the same data to multiple disks at the same time. The mirroring of data protects it against data loss within the database or within the recovery log.

multi-tailed
A disk connected to multiple nodes.

N

namespace
Space reserved by a file system to contain the names of its objects.

Network File System (NFS)
A protocol, developed by Sun Microsystems, Incorporated, that allows any host in a network to gain access to another host or netgroup and their file directories.

Network Shared Disk (NSD)
A component for cluster-wide disk naming and access.

NSD volume ID
A unique 16 digit hex number that is used to identify and access all NSDs.

node An individual operating-system image within a cluster. Depending on the way in which the computer system is partitioned, it may contain one or more nodes.

node descriptor
A definition that indicates how GPFS uses a node. Possible functions include: manager node, client node, quorum node, and nonquorum node

node number
A number that is generated and maintained by GPFS as the cluster is created, and as nodes are added to or deleted from the cluster.

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

node quorum with tiebreaker disks
A form of quorum that allows GPFS to run with as little as one quorum node available, as long as there is access to a majority of the quorum disks.

non-quorum node
A node in a cluster that is not counted for the purposes of quorum determination.

P

policy A list of file-placement and service-class rules that define characteristics and placement of files. Several policies can be defined within the configuration, but only one policy set is active at one time.

policy rule
A programming statement within a policy that defines a specific action to be preformed.

pool A group of resources with similar characteristics and attributes.

portability
The ability of a programming language to compile successfully on different operating systems without requiring changes to the source code.

primary GPFS cluster configuration server
In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data.

private IP address
A IP address used to communicate on a private network.

public IP address
A IP address used to communicate on a public network.

quorum node
A node in the cluster that is counted to determine whether a quorum exists.

Q

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

quota management
The allocation of disk blocks to the other

nodes writing to the file system, and comparison of the allocated space to quota limits at regular intervals.

R

Redundant Array of Independent Disks (RAID)
A collection of two or more disk physical drives that present to the host an image of one or more logical disk drives. In the event of a single physical device failure, the data can be read or regenerated from the other disk drives in the array due to data redundancy.

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

replication
The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target), and synchronizing the data in both locations.

rule A list of conditions and actions that are triggered when certain conditions are met. Conditions include attributes about an object (file name, type or extension, dates, owner, and groups), the requesting client, and the container name associated with the object.

S

SAN-attached
Disks that are physically attached to all nodes in the cluster using Serial Storage Architecture (SSA) connections or using fibre channel switches

secondary GPFS cluster configuration server
In a GPFS cluster, the node chosen to maintain the GPFS cluster configuration data in the event that the primary GPFS cluster configuration server fails or becomes unavailable.

Secure Hash Algorithm digest (SHA digest)
A character string used to identify a GPFS security key.

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

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

Small Computer System Interface (SCSI)
An ANSI-standard electronic interface that allows personal computers to communicate with peripheral hardware, such as disk drives, tape drives, CD-ROM drives, printers, and scanners faster and more flexibly than previous interfaces.

snapshot
A copy of changed data in the active files and directories of a file system with the exception of the inode number, which is changed to allow application programs to distinguish between the snapshot and the active files and directories.

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

stand-alone client
The node in a one-node cluster.

storage area network (SAN)
A dedicated storage network tailored to a specific environment, combining servers, storage products, networking products, software, and services.

storage pool
A grouping of storage space consisting of volumes, logical unit numbers (LUNs), or addresses that share a common set of administrative characteristics.

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

striping
A storage process in which information is split into blocks (a fixed amount of data) and the blocks are written to (or read from) a series of disks in parallel.

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

system storage pool
A storage pool containing file system control structures, reserved files, directories, symbolic links, special devices,

as well as the metadata associated with regular files, including indirect blocks and extended attributes The **system storage pool** can also contain user data.

T

token management
A system for controlling file access in which each application performing a read or write operation is granted some form of access to a specific block of file data. Token management provides data consistency and controls conflicts. Token management has two components: the token management server, and the token management function.

token management function
A component of token management that requests tokens from the token management server. The token management function is located on each cluster node.

token management server
A component of token management that controls tokens relating to the operation of the file system. The token management server is located at the file system manager node.

twin-tailed
A disk connected to two nodes.

U

user storage pool
A storage pool containing the blocks of data that make up user files.

V

virtual file system (VFS)
A remote file system that has been mounted so that it is accessible to the local user.

virtual node (vnode)
The structure that contains information about a file system object in an virtual file system (VFS).

Index

Special characters

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Program Number: 5724-N94
5765-G66

Printed in USA

GA76-0413-04