General Parallel File System

Concepts, Planning, and Installation Guide

Version 3 Release 2.1

GA76-0413-02

General Parallel File System

Concepts, Planning, and Installation Guide

Version 3 Release 2.1

GA76-0413-02

Note:
Before using this information and the product it supports, be sure to read the general information under “Notices” on page 107.

Third Edition (August 2008)

This edition applies to version 3, release 2, modification 1 of IBM General Parallel File System Multiplatform (product number 5724-N94), IBM General Parallel File System for POWER (product number 5765-G66), and to all subsequent releases and modifications until otherwise indicated in new editions. Technical changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

IBM welcomes your comments. A form for your comments may be provided at the back of this publication, or you may address your comments to the following:

International Business Machines Corporation
Department 58HA, Mail Station P181
2455 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States and Canada): 1+845+432-9405
FAX (Other Countries):
Your International Access Code +1+845+432-9405

IBMLink™ (United States customers only): IBMUSM10(MHVRCFS)
Internet e-mail: mhvrcfs@us.ibm.com

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this publication
- Page number or topic related to your comment

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* describes:

- The IBM® General Parallel File System™ (GPFS™) Multiplatform licensed program, 5724-N94
- The IBM GPFS for POWER™ licensed program, 5765-G66

This information includes 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 3.2.1 for AIX®, Linux®, and Windows®.

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

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

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

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

| To find out which version of GPFS is running on a particular Windows node, use the graphical user
| interface (GUI) and follow these steps:

| 1. Click **Control Panel**–>**Add or Remove Programs**
| 2. Click **IBM General Parallel File System** and choose **Click here for support information.**

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 describes the typographic conventions used in this information.

Table 1. Typographic conventions

Typographic 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 fail to specify a different keyword.
Italic	• *Italic* words or characters represent variable values that you must supply. • *Italics* are also used for publication titles and for general emphasis in text.
`Constant width`	All of the following are displayed in `constant width` typeface: • Displayed information • Message text • Example text • Specified text typed by the user • Field names as displayed on the screen • Prompts from the system • References to example text
[ ]	Brackets enclose optional items in format and syntax descriptions.
{ }	Braces enclose a list from which you must choose an item in format and syntax descriptions.
|	A vertical bar separates items in a list of choices. (In other words, it means ″or″)
<>	Angle brackets (less-than and greater-than) enclose the name of a key on the keyboard. For example, **<Enter>** refers to the key on your terminal or workstation that is labeled with the word Enter.
...	An ellipsis indicates that you can repeat the preceding item one or more times.
<Ctrl-*x*>	The notation **<Ctrl-*x*>** indicates a control character sequence. For example, **<Ctrl-*c*>** means that you hold down the control key while pressing **<c>**.
\	The continuation character is used in programming examples in this information for formatting purposes.

Prerequisite and related information

For updates to this information, see publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfsbooks.html.

For the latest support information, see the GPFS Frequently Asked Questions at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends and codes. You can use LookAt from the following locations to find IBM message explanations for Clusters software products:

- The Internet. You can access IBM message explanations directly from the LookAt Web site: http://www.ibm.com/systems/z/os/zos/bkserv/lookat/
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

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 by e-mail to: mhvrcfs@us.ibm.com

  Include the publication title and order number, and, if applicable, the specific location of the information you have comments on (for example, a page number or a table number).
- Fill out one of the forms at the back of this information and return it by mail, by fax, or by giving it to an IBM representative.

To contact the IBM cluster development organization, send your comments by e-mail to: cluster@us.ibm.com.

Summary of changes

The following sections summarize changes to the GPFS licensed program and the GPFS library for version 3, release 2, modification 1. 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.

Summary of changes
for GPFS Version 3, Release 2, Modification 1
as updated, August 2008

Changes to GPFS and to the GPFS library for version 3, release 2, modification 1 include:

- **New information**
  - GPFS for Windows Multiplatform, V3.2.1 supports the Windows Server 2003 R2 operating system running on 64-bit architectures (AMD x64 / EM64T). GPFS on Windows participates in a new or existing GPFS V3.2 cluster in conjunction with AIX and Linux (32- or 64-bit) operating systems.
  - Identity mapping between Windows and UNIX® user accounts is one of the key advancements delivered in GPFS for Windows Multiplatform. System administrators can explicitly match users and groups defined on UNIX with those defined on Windows. This allows users to 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. For more information about identity mapping, see the *General Parallel File System: Concepts, Planning, and Installation Guide* and the *General Parallel File System: Advanced Administration Guide*.
  - IBM has enhanced many of the details within GPFS to support the unique semantic requirements posed by Windows. These 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 long-standing capabilities provided on AIX and Linux operating systems.
  - GPFS for Windows Multiplatform provides the same core services to parallel and serial applications as are available on AIX and Linux operating systems. GPFS allows parallel applications simultaneous access to the same files, or different files, from any node that has the GPFS file system 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 operating systems.

    The following commands have been updated for Windows:

    - **mmchfs** to add the **-t** *DriveLetter* option
    - **mmcrfs** to add the **-t** *DriveLetter* option
    - **mmlsfs** to add the **-t** option to display the Windows drive letter
    - **mmmount** to add the *DefaultDriveLetter* and *DriveLetter* parameters
    - **mmumount** to add the *DefaultDriveLetter* and *DriveLetter* parameters

    With few exceptions, the commands supported on the Windows operating system are identical to the commands supported on other GPFS platforms. For a list of unsupported commands, see the *General Parallel File System: Concepts, Planning, and Installation Guide*.
  - GPFS for Windows Multiplatform, V3.2.1 does not support or has restricted support for some features. For a complete list of these limitations, see the *General Parallel File System: Concepts, Planning, and Installation Guide*.
- **Changed information:**

  Minor editorial updates marked by a vertical line to the left of the text.
- **Deleted information:**

  There has been no information deleted from the GPFS library for GPFS V3.2.1.

Chapter 1. Introducing General Parallel File System

IBM's General Parallel File System (GPFS) provides file system services to parallel and serial applications. GPFS allows parallel applications simultaneous access to the same files, or different files, from any node which has the GPFS file system mounted while managing a high level of control over all file system operations.

GPFS is particularly appropriate in an environment where the aggregate peak need for data bandwidth exceeds the capability of a distributed file system server.

GPFS allows users shared file access within a single GPFS cluster and across multiple GPFS clusters. A GPFS cluster consists of:

- AIX nodes, Linux nodes, Windows nodes, or a combination thereof (see "GPFS cluster configurations" on page 7). A node can be:
  - An individual operating system image on a single computer within a cluster.
  - A system partition that contains an operating system. Some IBM System p5™ and IBM System p™ machines allow multiple system partitions, each of which is considered to be a node within a GPFS cluster.
- Network shared disks (NSDs) created and maintained by the NSD component of GPFS
  - All disks used by GPFS must first be given a globally-accessible NSD name.
  - The GPFS NSD component provides a method for cluster-wide disk naming and access.
  - On Linux machines running GPFS, you may give an NSD name to:
    - Physical disks
    - Logical partitions of a disk
    - Representations of physical disks (such as LUNs)
  - On AIX machines running GPFS, you may give an NSD name to:
    - Physical disks
    - Virtual shared disks
    - Representations of physical disks (such as LUNs)
- A shared network for GPFS communications allowing a single network view of the configuration. A single network, a LAN or a switch, is used for GPFS communication, including the NSD communication.

The strengths of GPFS

GPFS is a powerful file system that provides global namespace, shared file system access among GPFS clusters, simultaneous file access from multiple nodes, high recoverability and data availability due to replication, the ability to make certain changes while a file system is mounted, and simplified administration that is similar to existing UNIX systems.

For more information, see the following:

Shared file system access among GPFS clusters

GPFS allows users shared access to files in either the cluster where the file system was created or other GPFS clusters. Each site in the network is managed as a separate cluster, while allowing shared file system access. 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.

Note: If you use a cipher, the data will be encrypted for transmissions. However, if you set the **cipherlist** keyword of the **mmauth** command to **AUTHONLY**, only authentication will be used for data transmissions and data will not be encrypted.

GPFS shared file system access provides for:

- The ability of the cluster granting access to specify multiple security levels, up to one for each authorized cluster.
- 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 can not 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.

Note: The pair of public and private security keys provided by GPFS are similar to 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 on shared file system access, see the *GPFS: Advanced Administration Guide*.

Improved system performance

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

- Allowing multiple processes or applications on all nodes in the 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. One disk is no more active than another.
- Supporting very large file and file system sizes.
- Allowing concurrent reads and writes from multiple nodes. This is a key concept in parallel processing.
- Allowing for distributed token (lock) management. Distributing token management reduces system delays associated with a lockable object waiting to obtaining a token. Refer to “Managing distributed tokens” on page 24 and “High recoverability and increased data availability” on page 3 for additional information on token management.
- Allowing for the specification of different 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 [1]

1. Dominique Heger, Gautam Shah: General Parallel File System (GPFS v1.4) for AIX Architecture and Performance, November 2001

To exploit disk parallelism when reading a large file from a single-threaded application, whenever it can recognize a pattern, GPFS prefetches data into its buffer pool, issuing I/O requests in parallel to as many disks as necessary to achieve the bandwidth of which the switching fabric is capable. GPFS recognizes sequential, reverse sequential, and various forms of strided access patterns [1].

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 10, "GPFS architecture," on page 79.

High recoverability and increased data availability

GPFS failover support allows you to organize your hardware into *failure groups*. A failure group is a set of disks that share a common point of failure that could cause them all to become simultaneously unavailable. 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. GPFS maintains each instance of replicated data and metadata on disks in different failure groups. 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. Replication allows you to set different levels of protection for each file or one level for an entire file system. Since replication uses additional disk space and requires extra write time, you might want to consider replicating only file systems that are frequently read from but seldom written to. 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 recoverability parameters" on page 36.

GPFS is a logging file system that creates separate logs for each node. These logs record the allocation and modification of metadata aiding in fast recovery and the restoration of data consistency in the event of node failure. 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 have been specified. This replication feature can be used in conjunction with other GPFS capabilities to maintain one replica in a geographically separate location which provides some capability for surviving disasters at the other location. For further information on failure groups, see "NSD creation considerations" on page 25. 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 indicate 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 lessened 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 for 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, you must switch quorum semantics to node quorum. For additional information, refer to “Quorum” on page 15.

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 on the 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:

- 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
- 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 UNIX file system interfaces allowing most applications to execute without modification or recompiling. UNIX file system utilities are also supported by GPFS. That is, users can continue to use the UNIX commands they have always used for ordinary file operations (see Chapter 12, “Considerations for GPFS applications,” on page 103). The only unique commands are those for administering the GPFS file system.

GPFS administration commands are similar in name and function to UNIX file system commands, with one important difference: *the GPFS commands operate on multiple nodes*. A single GPFS command performs a file system function across the entire 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 are designed to 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 93.

The basic GPFS structure

GPFS is a clustered file system defined over a number of nodes. On each node in the cluster, GPFS consists of: administration commands, a kernel extension, a multithreaded daemon, and for Linux nodes, an open source portability layer.

For more information, see the following topics:

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

For a detailed discussion of GPFS, see Chapter 10, “GPFS architecture,” on page 79.

GPFS administration commands

Most GPFS administration tasks can be performed from any node running GPFS. See the individual commands as documented 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. In this way, GPFS appears to applications as just another file system. The GPFS kernel extension will either satisfy these requests using resources which are already available in the system, or send a message to the GPFS daemon to complete the request.

The GPFS daemon

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

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. The daemon also communicates with instances of the daemon on other nodes to coordinate configuration changes, recovery and parallel updates of the same data structures. Specific functions that execute on the daemon include:

1. Allocation of disk space to new files and newly extended files. This is done in coordination with the *file system manager*.
2. Management of directories including creation of new directories, insertion and removal of entries into existing directories, and searching of directories that require I/O.
3. Allocation of appropriate locks to protect the integrity of data and metadata. Locks affecting data that may be accessed from multiple nodes require interaction with the token management function.
4. Disk I/O is initiated on threads of the daemon.
5. User security and quotas are also managed by the daemon 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 disk, each of its NSD servers must have physical access to the same LUN. 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 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 85.

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

The default order of access used in disk discovery:

1. Local **/dev** block device interfaces for virtual shared disk, 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 24 and “NSD disk discovery” on page 92.

The GPFS open source portability layer

For Linux nodes running GPFS, you must build custom portability modules based on your particular hardware platform and Linux distribution to enable communication between the Linux kernel and the GPFS kernel modules. See “Building your GPFS portability layer” on page 45.

GPFS cluster configurations

GPFS can be configured in a variety of ways. This topic includes illustrations of several configurations.

A subset of these configurations includes:

1. Configurations where all disks are SAN-attached to all nodes in the cluster and the nodes in the cluster are either all Linux (refer to Figure 1) or all AIX. For the latest hardware that GPFS has been tested with, see the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html.

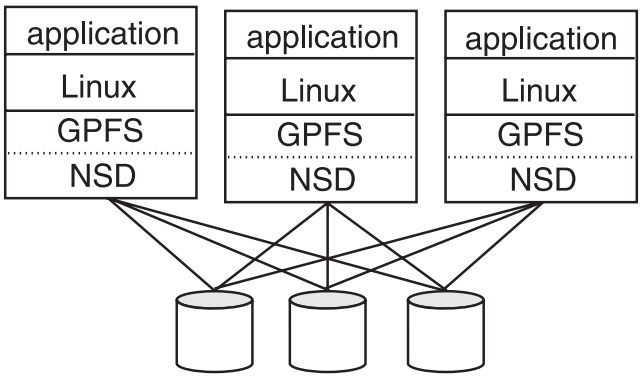


Figure 1. A Linux-only cluster with disks that are SAN-attached to all nodes

2. A Linux-only cluster consisting of xSeries®, IBM System p5, IBM System p, or eServer™ processors with an NSD server attached to the disk (refer to Figure 2). Nodes that are not directly attached to the disk have remote data access over the local area network (either Ethernet or Myrinet) to the NSD server. An NSD server with direct Fibre Channel access to the disks can also be defined. Any nodes directly attached to the disk will not access data through the NSD server. This is the default behavior, which can be changed with the **useNSDserver** file system mount option.

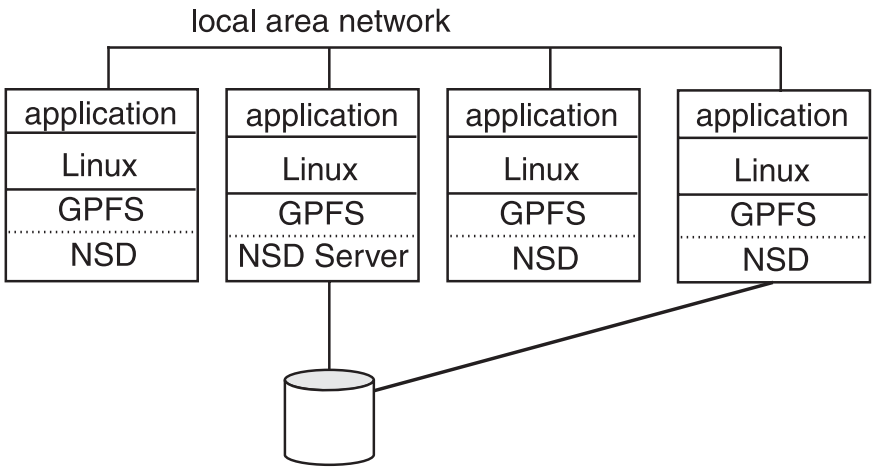


Figure 2. A Linux-only cluster with an NSD server

3. An AIX and Linux cluster with an NSD server (refer to Figure 3 on page 8). Nodes not directly attached to the disk have remote access over the local area network to the NSD server.

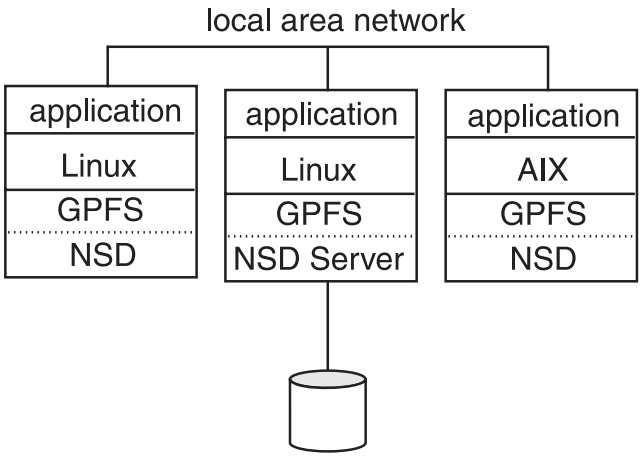


Figure 3. An AIX and Linux cluster with an NSD server

4. An AIX and Linux cluster with an NSD server (refer to Figure 4). AIX nodes with Recoverable Virtual Shared Disk (RVSD) and IBM High Performance Switch (HPS) connections to the disk server access data through that path. Linux nodes or other AIX nodes with no RVSD switch access have remote data access over the LAN to the NSD server. You can also define additional NSD servers with access to the disk through RVSD and the switch.

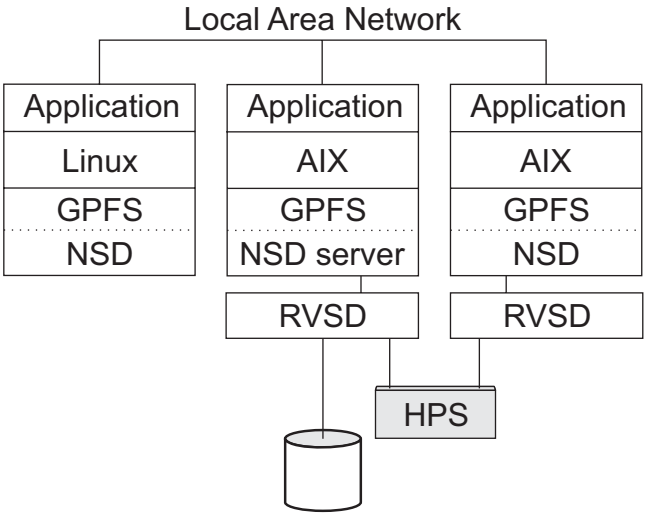


Figure 4. An AIX and Linux cluster providing remote access to disks through the High Performance Switch (HPS) for the AIX nodes and a LAN connection for the Linux nodes

5. An AIX and Linux cluster that provides remote access to disks through multiple NSD servers (refer to Figure 5 on page 9). In this configuration:
   - Internode communication uses the LAN
   - All disk access passes through NSD servers
     - NSD servers connect the disk to the Linux nodes
     - NSD servers use RVSD to connect the disk to the AIX nodes

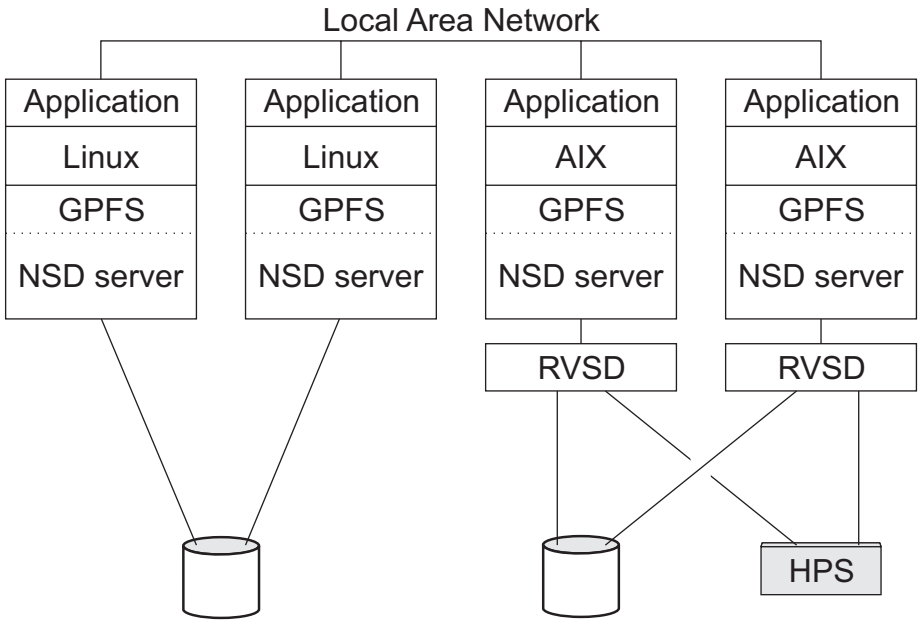


Figure 5. An AIX and Linux cluster that provides remote access to disks through multiple NSD servers

6. An AIX cluster with an NSD server (refer to Figure 6). Nodes not directly attached to the disk have remote access over the HPS to the NSD server.

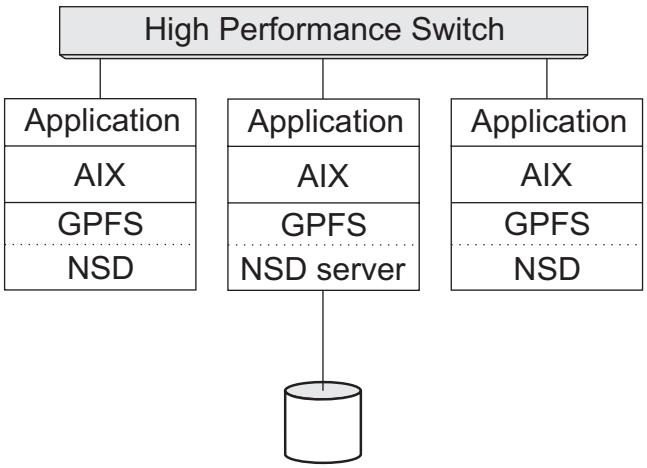


Figure 6. An AIX cluster with an NSD server

7. Shared file system access among multiple GPFS clusters (refer to Figure 7 on page 10). The GPFS clusters sharing file system access may be any supported configuration.

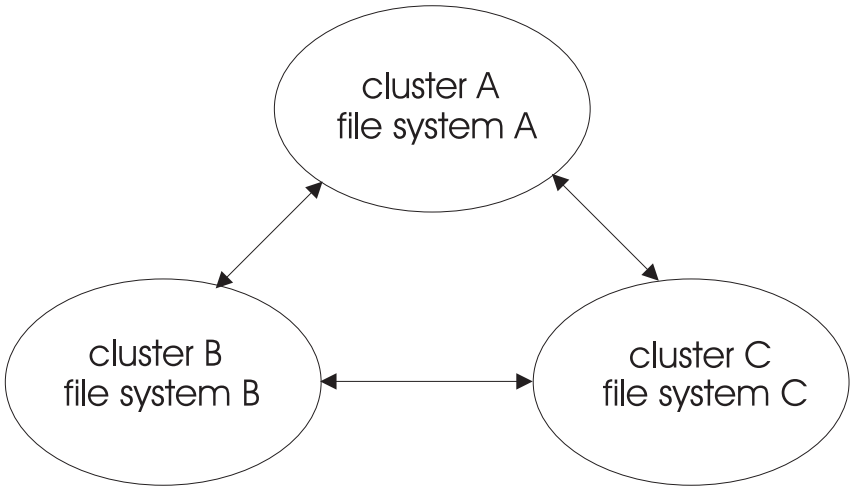


Figure 7. GPFS clusters providing shared file system access

For the latest list of supported cluster configurations, please see the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html.

Interoperable cluster requirements

Consult the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html for any changes to requirements and currently tested:

1. Hardware configurations
2. Software configurations
3. Cluster configurations

Prior to GPFS 3.2, upgrading your system to a new version of GPFS required shutting down GPFS and upgrading all nodes before you could restart GPFS. However, if you are upgrading a GPFS 3.1 cluster to GPFS 3.2, you can perform a rolling upgrade with a limited form of backward compatibility. 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 3.2 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 all** (or **mmchfs compat**). Once this is done, you can create new file systems and use such new features as Persistent Reserve (PR) and multiple NSD servers.

Limited backward compatibility allows you to temporarily operate your cluster with a mixture of old and new nodes. In addition, GPFS requires backward compatibility for multicluster environments. With backward compatibility, the administrator should be able to upgrade the local cluster while still allowing mounts from remote nodes in other clusters that have not been upgraded yet. For additional information, refer to Chapter 7, “Migration, coexistence and compatibility,” on page 59.

These configuration requirements apply to an interoperable GPFS cluster:

- All file systems defined on versions of GPFS prior to version 2.3 must be exported from their old cluster definition and re-imported into a newly created GPFS 3.2 cluster. Cluster configuration dependencies and setup changed significantly in GPFS version 2.3. See “Migrating to GPFS 3.2 from GPFS 2.2 or earlier releases of GPFS” on page 60.
- All nodes serving a set of NSDs must be on a homogenous set of Linux or AIX nodes. An NSD cannot be split between operating system types. See Figure 5 on page 9.

- For most disk subsystems, all nodes accessing a SAN-attached disk (LUN) must use the same operating system. Most disk subsystems do not allow you to have Linux nodes and AIX nodes nodes attached to the same LUN. Refer to the information supplied with your specific disk subsystem for details about supported configurations.
- Your cluster can have a mix of nodes with GPFS 3.1 (RDMA not available), GPFS 3.2 with RDMA configured, and GPFS 3.2 without RDMA configured. Only the GPFS 3.2 nodes with RDMA configured will use RDMA for data transfer between the NSD client and server.

Note: InfiniBand RDMA for Linux X86_64 is supported only on GPFS V3.2 Multiplatform. For the latest support information, see the GPFS Frequently Asked Questions at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html

Chapter 2. 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 have the opportunity to specify parameters based on the expected size of the files or allow the default values to take effect. These parameters define the disks for the file system and how data will be written to them.

Planning for GPFS includes:

Hardware requirements

There are three steps to consider to meet the hardware requirements. This topic will describe those steps as well as provide references for more information.

1. Consult the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html for latest list of:
   - Supported hardware
   - Tested disk configurations
   - Maximum cluster size
2. 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 "NSD creation considerations" on page 25 for additional information.
3. Since GPFS passes a large amount of data between its daemons, it is suggested that you configure a dedicated high speed network supporting the IP protocol when you are using GPFS:
   - With NSD disks configured with servers providing remote disk capability
   - Multiple GPFS clusters providing remote mounting of and access to GPFS file systems

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

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

Software requirements

Part of GPFS planning includes understanding the latest software requirements.

Consult the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html for the latest list of:

- Linux distributions
- Linux kernel versions
- AIX environments
- Windows environments (only Windows Server 2003 R2 x64 is supported)
- OpenSSL levels

Note: When multiple clusters are configured to access the same GPFS file system OpenSSL is used to authenticate and check authorization for all network connections. In addition, if you use a cipher, data will be encrypted for transmissions. However, if you set the **cipherlist** keyword of the **mmauth** command to **AUTHONLY**, only authentication will be used for data transmissions and data will not be encrypted.

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 fast recoverability from failures.

- At the file system level, consider replication through the metadata and data replication parameters. See "File system recoverability parameters" on page 36.
- 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 "NSD creation considerations" on page 25.

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 17
3. "Reduced recovery time using Persistent Reserve" on page 20

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 failing 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. For further information on 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 being modified on the failing node. Recovery involves rebuilding metadata structures which may have been under modification at the time of the failure. If the failing node is the file system manager for the file system, the delay will be longer and proportional to the activity on the file system at the time of failure. However, administrative intervention will not be needed.

Note: If **usePersistentReserve** is enabled, GPFS prevents the continuation of I/O from a failing node by fencing the failed node using Persistent Reserve (SCSI-3 protocol). Persistent Reserve allows the

failing node to recover faster. 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 20.

Quorum

During node failure situations, quorum needs to be maintained in order to recover the failing nodes. 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 17 for additional information).

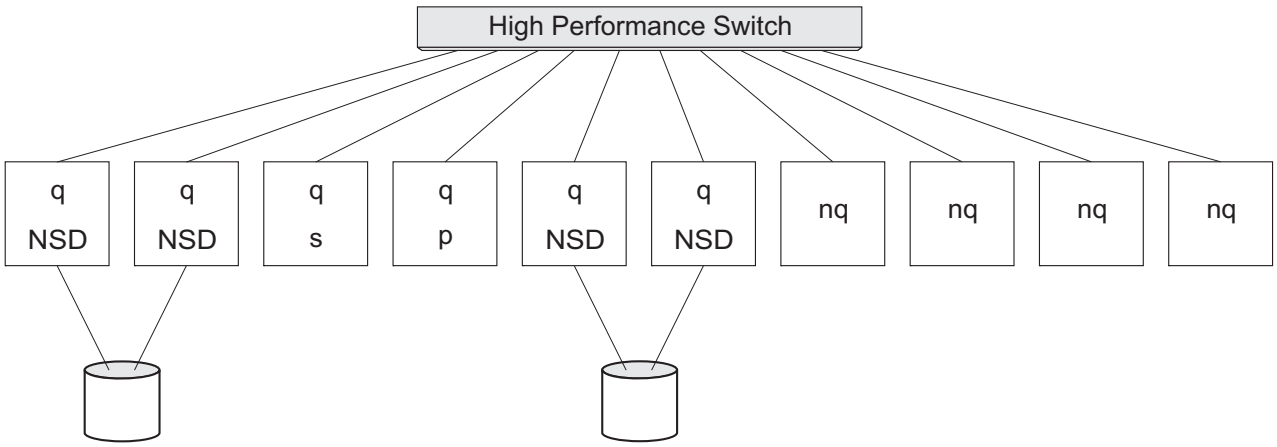
GPFS quorum must be maintained within the cluster for GPFS to remain active. If the quorum semantics are broken, GPFS performs recovery in an attempt to achieve quorum again. GPFS can use one of two methods for determining quorum:

- 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.
- GPFS does not limit the number of quorum nodes.

For example, in Figure 8, there are six quorum nodes. In this configuration, GPFS remains active as long as there are four quorum nodes available.



q - quorum node

NSD - NSD server

s - secondary cluster configuration server

p - primary cluster configuration server

nq - non-quorum node

Figure 8. GPFS configuration utilizing node quorum

Node quorum with tiebreaker disks: 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 9 on page 17). Switching to quorum with tiebreaker disks is accomplished by indicating a list of one to three disks to use on the **tiebreakerDisks** parameter on the **mmchconfig** command.

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. You should include the primary and secondary cluster configuration servers as quorum nodes.
3. You may have an unlimited number of non-quorum nodes.

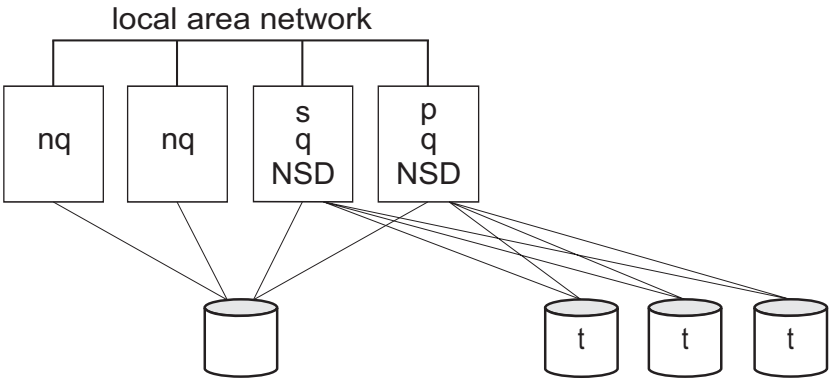
Changing quorum semantics:

1. If you exceed eight quorum nodes, you must disable node quorum with tiebreaker disks and restart GPFS daemon using the default node quorum configuration. To disable node quorum with tiebreaker disks:
   a. Shutdown the GPFS daemon by issuing **mmshutdown -a** on all nodes.
   b. Change quorum semantics by issuing **mmchconfig tiebreakerdisks=no**.
   c. Add quorum nodes.
   d. Restart the GPFS daemon by issuing **mmstartup -a** 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. Shutdown the GPFS daemon by issuing **mmshutdown -a** on all nodes.
   b. Delete the quorum nodes.
   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. Restart the GPFS daemon by issuing **mmstartup -a** 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.
- Tiebreaker disks must be defined through the **mmcrnsd** command.
- Tiebreaker disks must use one of following attachments to the quorum nodes:
  – fibre-channel SAN
  – IP SAN
  – virtual shared disks

In Figure 9 on page 17 GPFS remains active with the minimum of a single available quorum node and two available tiebreaker disks.



p - primary cluster configuration server

s - secondary cluster configuration server

q - quorum node

nq - non-quorum node

NSD - NSD server

t - tiebreaker disk

Figure 9. GPFS configuration utilizing 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.

| **Note:** Windows nodes cannot be selected as quorum nodes.

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 replication as provided by a Redundant Array of Independent Disks (RAID) device

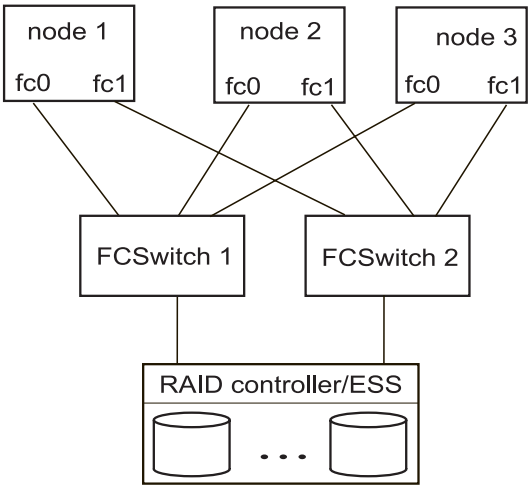


Figure 10. RAID/ESS Controller twin-tailed in a SAN configuration

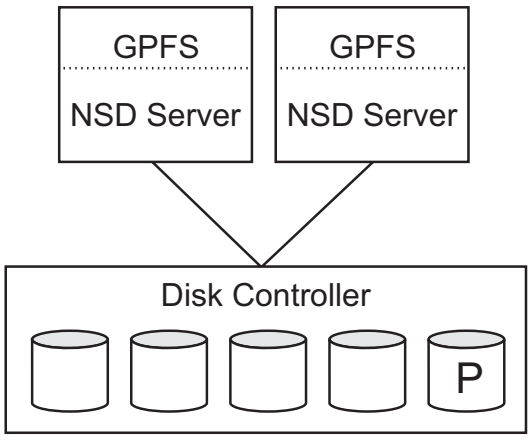


Figure 11. GPFS configuration specifying multiple NSD servers connected to a common disk controller utilizing RAID5 with four data disks and one parity disk

- Utilizing the GPFS data and metadata replication features (see “High recoverability and increased data availability” on page 3) along with the designation of failure groups (see “NSD creation considerations” on page 25)

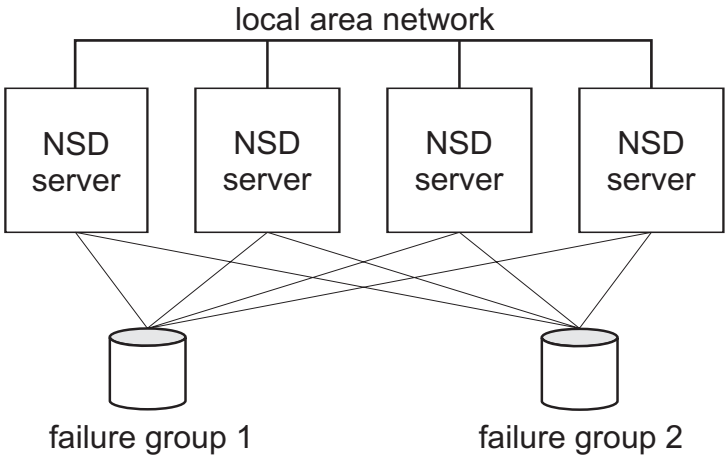


Figure 12. GPFS utilizes failure groups to minimize the probability of a service disruption due to a single component failure

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

In the event of failure of a path to the disk:

- If a virtual shared disk server goes down and GPFS reports a disk failure, follow the instructions in the *RSCT for AIX 5L Managing Shared Disks* manual for the level of your system to check the state of the virtual shared disk path to the disk.
- If a SAN failure removes the path to the disk and GPFS reports a disk failure, follow the directions supplied by your storage vendor to distinguish a SAN failure from a disk failure.

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

- 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 NSD servers for the disks. 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 falls back to local disk access. This is the default behavior, which can be changed with the **-o useNSDserver** file system mount option on the **mmchfs**, **mmmount**, **mmremotefs**, and **mount** commands.
- Using the Multiple Path I/O (MPIO) feature of AIX to define alternate paths to a device 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. All Small Computer System Interface (SCSI) Self Configured SCSI Drive (SCSD) disk drives are automatically configured as MPIO devices. Other devices can be supported, providing the device driver is compatible with the MPIO implementation in AIX. For more information about MPIO, see the:
  - *AIX 5L Version 5.3 System Management Concepts: Operating System and Devices* book and search on *Multi-path I/O*.
  - *AIX 5L Version 5.3 System Management Guide: Operating System and Devices* book and search on *Multi-path I/O*.

- Use Subsystem Device Driver (SDD) or Subsystem Device Driver Path Control Module (SDDPCM) to give the AIX host the ability to access multiple paths to a single LUN within an Enterprise Storage Server® (ESS). This ability to access a single logical unit number (LUN) on multiple paths allows for a higher degree of data availability in the event of a path failure. Data can continue to be accessed within the ESS as long as there is at least one available path. Without one of these installed, you will lose access to the LUN in the event of a path failure. For additional information about:
  - SSD, refer to http://www.ibm.com/server/storage/support/software/sdd/
  - SDDPCM, refer to http://www.ibm.com/support/docview.wss?uid=ssg1S4000201

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 will attempt to setup PR on all the PR capable disks. All subsequent NSDs will be created with PR enabled if they are PR capable. However, PR will only be supported in the home cluster. Therefore, remote mounts must access PR disks through an NSD server that is in the home cluster.

GPFS cluster creation considerations

To create GPFS clusters, issue the **mmcrcluster** command. This topic describes the GPFS cluster creation options.

Table 2 details:

- The GPFS cluster creation options provided by the **mmcrcluster** command
- How to change the options
- What the default values are for each option

Note: Refer to the *GPFS: Advanced Administration Guide* for information on accessing GPFS file systems in remote clusters and large cluster administration.

Table 2. GPFS cluster creation options

Cluster option	Command to change the option	Default value
"Nodes in your GPFS cluster" on page 21	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 21	**mmchnode**	Client
Node designation: Quorum or non-quorum, see "Nodes in your GPFS cluster" on page 21	**mmchnode**	Non-quorum
Primary cluster configuration server, see "GPFS cluster configuration servers" on page 22	**mmchcluster**	None

Table 2. GPFS cluster creation options (continued)

Cluster option	Command to change the option	Default value
Secondary cluster configuration server, see “GPFS cluster configuration servers” on page 22	**mmchcluster**	None
“Remote shell command” on page 22	**mmchcluster**	**/usr/bin/rsh**
“Remote file copy command” on page 23	**mmchcluster**	**/usr/bin/rcp**
“Cluster name” on page 23	**mmchcluster**	The node name of the primary GPFS cluster configuration server
GPFS administration adapter port name, see “GPFS node adapter interface names”	**mmchnode**	Same as the GPFS communications adapter port name
GPFS communications adapter port name, see “GPFS node adapter interface names”	**mmchnode**	None
“User ID domain for the cluster” on page 23	**mmchconfig**	The name of the GPFS cluster
“Starting GPFS automatically” on page 23	**mmchconfig**	No
“Cluster configuration file” on page 23	**mmchconfig**	None
“Managing distributed tokens” on page 24	**mmchconfig**	Yes

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. GPFS permits the administrator to 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**, **mmchcnode**, or **mmcrcluster** command.

Nodes in your GPFS cluster

When you create your GPFS cluster you must provide a file containing a list of node descriptors, one per line, for each node to be included in the cluster. GPFS stores this information on the “GPFS cluster configuration servers” on page 22. Each descriptor must be specified in the form:

```
NodeName:NodeDesignations:AdminNodeName
```

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)

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 special functions of the file system manager consume extra processing time. See "The file system manager" on page 80. The default is to *not* have the node included in the pool.

  In general, small systems do not need multiple nodes dedicated for the file system manager. However, if you are running large parallel jobs, threads scheduled to a node performing these functions may run slower. As a guide, in a large system there should be 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 non-quorum. You must designate at least one node as a quorum node. It is recommended that you designate 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 15.

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.

You must follow these rules when creating your GPFS cluster:

- While a node may mount file systems from multiple clusters, the node itself may only be added to a single cluster through either the **mmcrcluster** command or the **mmaddnode** command.
- The default quorum type is node quorum. To enable node quorum with tiebreaker disks, you must issue the **mmchconfig** command.
- The node must be available for the command to be successful. If any of the nodes listed are not available when the command is issued, a message listing those nodes is displayed. You must correct the problem on each node, create a new input file containing the failed nodes only, and issue the **mmaddnode** command to add those nodes.

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.

Attention: If your primary server fails and you have not designated a secondary server, the GPFS cluster configuration data files are inaccessible and any GPFS administration commands that are issued, fail. 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 93.

Remote shell command

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

If you choose to designate the use of a different remote shell command on either the **mmcrcluster** or the **mmchcluster** command, you must specify the fully qualified pathname for the program to be used by GPFS. You must also ensure:

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

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

Note: If a Windows node is in the cluster, **rsh** cannot be used.

Remote file copy command

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

If you choose to designate the use of a different remote file copy command on either the **mmcrcluster** or the **mmchcluster** command, you must specify the fully-qualified pathname for the program to be used by GPFS. You must also ensure:

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

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

Note: If a Windows node is in the cluster, **rcp** cannot be used.

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* at www.ibm.com/servers/eserver/clusters/library/wp_aix_lit.html

Starting GPFS automatically

Specify whether to start GPFS automatically on all nodes in the cluster whenever they come up with the **autoload** attribute. The default is for GPFS to *not* automatically start GPFS on all nodes. You may change this by specifying the **autoload=yes** on the **mmchconfig** command. This eliminates the need to start GPFS by issuing the **mmstartup** command when a node comes back up.

Cluster configuration file

GPFS provides you with default configuration options that may generically apply to most systems. You may:

- Accept the system defaults at cluster creation time and tune your system after the cluster is created by issuing the **mmchconfig** command (see Chapter 8, “Configuring and tuning your system for GPFS,” on page 67). This is the suggested method.
- Experienced users may use a customized cluster configuration file on the **mmcrcluster** command. For detailed information, please see the *GPFS: Administration and Programming Reference*.

Managing distributed tokens

GPFS implements distributed locking using token-based lock management. In GPFS, the **distributedTokenServer** option of the **mmchconfig** command allows you to distribute the token server workload over multiple token manager nodes in a cluster. Distributing token management among file system manager nodes reduces system delays associated with a lockable object waiting to obtaining a token.

When the file system is mounted initially, the file system manager is the only token server. However, when the number of external mounts reaches a threshold, the file system manager appoints additional manager nodes as token servers.

Note:

1. The total number of token manager nodes depends on the number of manager nodes you defined in the cluster.
2. If you designated only one node as a manager node, you can have only one token server.
3. Once the token state has been distributed, it remains distributed until all external mounts have gone away.
4. The **maxFilesToCache** and **maxStatCache** parameters are indirectly affected by multiple token manager nodes as distributing tokens across multiple nodes could allow more tokens than if you only had one token server.
5. Refer to the *General Parallel File System: Advanced Administration Guide* for details on distributed token managers.

Disk considerations

Proper planning for your GPFS installation includes ensuring the disk considerations listed in this topic.

Do the following:

- 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. With GPFS 2.3 or later, you may have up to 268 million disks in the file system.

  Note:

  1. The actual number of disks in your system may be constrained by products other than the version of GPFS installed on your system.
  2. With file systems created with GPFS 2.3 or later, the theoretical limit on the maximum number of disks in a file system has been increased from 4096 to approximately 268 million. However, the actual limit enforced by the current version of GPFS is 2048. It can be increased if necessary (please contact IBM to discuss increasing the limit).
- Ensure that you have sufficient free disk space.

  If your system contains AIX nodes with NSDs created on existing virtual shared disks, sufficient free disk space needs to be kept in **/var** for the correct operation of the IBM Recoverable Virtual Shared Disk component. While GPFS does not use large amounts of **/var**, this component requires several megabytes to correctly support GPFS recovery. Failure to supply this space on all nodes may appear as a hang in your GPFS file system when recovering failed nodes.

- Ensure that you have sufficient connectivity (adapters and buses) between disks and network shared disk servers.
- Decide how your disks will be connected. Supported types of disk connectivity include:
  1. All disks are SAN-attached to all nodes in all clusters which access the file.

     In this configuration, every node sees the same disk simultaneously and has a corresponding disk device entry in **/dev**.
  2. Each disk is 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 or virtual 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 if you will use storage pools to manage your disks.

  Storage pools allow you to manage your file system's storage in groups. You may now 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:

  – 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

  – Snapshot management

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

Disk considerations include:

1. "NSD creation considerations"
2. "NSD server considerations" on page 28
3. "File system descriptor quorum" on page 29

NSD creation considerations

You must prepare each physical disk you intend to use with GPFS as an NSD through the **mmcrnsd** command. NSDs can be created on:

- Physical disks
  – An hdisk or vpath on AIX
  – A block disk device or a disk partition on Linux

Note: An NSD cannot be created on disks that are attached to Windows (that is, a Windows node cannot be specified as an NSD server).

- Virtual shared disks:
  - An RSCT peer domain virtual shared disk, see *Reliable Scalable Cluster Technology: Managing Shared Disks*

Note: When you are using the High Performance Switch (HPS) in your configuration it is suggested you process your disks in two steps:

1. Create virtual shared disks on each physical disk through the **mmcrvsd** command.
2. Using the rewritten disk descriptors from the **mmcrvsd** command, create NSDs through the **mmcrnsd** command.

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

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 SAN-attached disks or virtual shared disks. If a server node is specified, *DiskName* must be the **/dev** name for the disk device of the first NSD server node defined in the server list. See the Frequently Asked Questions at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/ gpfs_faqs/ gpfsclustersfaq.html for the latest supported disk types.

In the AIX environment, GPFS provides the **mmcrvsd** command to ease configuration of virtual shared disks. This command allows you to configure a virtual shared disk and make it accessible to nodes connected over a High Performance Switch. In addition, you can use the output disk descriptor file from the **mmcrvsd** command as input to the **mmcrnsd** command.

Note: The virtual shared disk names listed in output disk descriptor file appear as **/dev** block devices on switch attached nodes.

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. The defined NSD will preferentially use the first server on the list. If the first server is not available, the NSD will use 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 ignored by the **mmcrnsd** command, and is passed unchanged to the output descriptor file produced by the **mmcrnsd** command. 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 for storage pools other than the system storage pool.
- **metadataOnly** – Indicates that the disk contains metadata and does not contain data.
- **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, and can be 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 that the NSD is assigned to. 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
  - 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 3 details the use of disk descriptor information by the GPFS disk commands:

Table 3. Disk descriptor usage for the GPFS disk commands

	mmcrnsd	mmcrvsd	mmchnsd	mmchdisk	mmcrfs	default value
Disk name	X	X	X	X	X	none
Server list	X	X	X	NA	NA	none
Disk usage	X	X	NA	X	X	dataAndMetadata for system storage pool Otherwise, dataOnly for all other storage pools
Failure group	X	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	X	NA	NA	NA	**gpfs***NN***nsd**, where *NN* is a unique nonnegative integer not used in any prior NSD
Storage pool	X	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 80.
- 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, the GPFS code 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

There is a structure in GPFS called the *file system descriptor* that is initially written to every disk in the file system, but is replicated on a subset of the disks as changes to the file system occur, such as adding or deleting disks. Based on the number of failure groups and disks, GPFS creates between one and 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 is decided how many replicas to create, GPFS picks disks to hold the replicas, so that all replicas will be 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 may modify the subset to rebalance the file system descriptors across the failure groups. The subset can be found by issuing the **mmlsdisk -L** command.

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

- If there are at least five different failure groups, GPFS will be able to tolerate a loss of two of the five groups. If disks out of three different failure groups are lost, the file system descriptor may become inaccessible due to the loss of the majority of the replicas.
- If there are at least three different failure groups, GPFS will be able to tolerate a loss of one of the three groups. If disks out of two different failure groups are lost, the file system descriptor may become inaccessible due to the loss of the majority of the replicas.

- If there are fewer than three failure groups, a loss of one failure group may make the descriptor inaccessible.

  If the subset consists of three disks and there are only two failure groups, one failure group must have two disks and the other failure group has one. In a scenario that causes one entire failure group to disappear all at once, if the half of the disks that are unavailable contain the single disk that is part of the subset, everything stays up. The file system descriptor is moved to a new subset by updating the remaining two copies and writing the update to a new disk added to the subset. But if the downed failure group contains a majority of the subset, the file system descriptor cannot be updated and the file system has to be force unmounted.

  Introducing a third failure group consisting of a single disk that is used solely for the purpose of maintaining a copy of the file system descriptor can help prevent such a scenario. You can designate this disk by using the **descOnly** designation for disk usage on the disk descriptor. With the **descOnly** designation, the disk does not hold any of the other file system data or metadata and can be as small as 4 MB. See *NSD creation considerations* in *GPFS: Concepts, Planning, and Installation Guide* and *Establishing disaster recovery for your GPFS cluster* in *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 at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html.

Your GPFS file system is created by issuing the **mmcrfs** command. Table 4 details the file system creation options specified on the **mmcrfs** command, which options can be changed later with the **mmchfs** command, and what the default values are.

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 4. File system creation options

Options	mmcrfs	mmchfs	Default value
Device name of the file system See “Device name of the file system” on page 32.	X	X	none
-D {**nfs4** | **posix**} semantics for a ’deny-write open lock’ See “NFS V4 ’deny-write open lock’” on page 33.	X	X	**posix**
DiskDesc for each disk in your file system See “Disks for your file system” on page 33.	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 32.	X	Issue the **mmadddisk** or **mmdeldisk** command to add or delete a file that contains a list of disk descriptors.	none

Table 4. File system creation options (continued)

Options	mmcrfs	mmchfs	Default value
-A {**yes** | **no** | **automount**} to determine when to mount the file system. See “Deciding how the file system is mounted” on page 33.	X	X	yes
-B *BlockSize* to set the data block size: **16K**, **64K**, **256K**, **512K**, **1M**, **2M**, or **4M**. See “Block size” on page 33.	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 35.	X	X	yes
-j {**cluster** | **scatter**} to determine the block allocation map type. See “Block allocation map” on page 35.	X	NA	See “Block allocation map” on page 35.
-k {**posix** | **nfs4** |**all**} to determine the authorization types supported by the file system. See “File system authorization” on page 35.	X	X	**posix**
-K {**no** | **whenpossible** | **always**} to enforce strict replication. See “Strict replication” on page 35.	X	X	**whenpossible**
-L *LogFileSize* to specify the size of the internal log file. See “GPFS recovery logs” on page 83.	X	NA	**4 MB**
-m *DefaultMetadataReplicas* See “File system recoverability parameters” on page 36.	X	X	1
-M *MaxMetadataReplicas* See “File system recoverability parameters” on page 36.	X	This value cannot be changed.	2
-N *NumInodes* to determine the maximum number of files in the file system. See “Maximum number of files” on page 37.	X	Use the **-F** option.	file system size/1 MB
-n *NumNodes* that will mount the file system. See “Number of nodes mounting the file system” on page 37.	X	This value cannot be changed after the file system has been created.	32

Table 4. File system creation options (continued)

Options	mmcrfs	mmchfs	Default value
-o *MountOptions* to be passed to the **mount** command. See “Assign mount command options” on page 37.	NA	X	none
-Q {**yes** | **no** } to activate quota. See Activate quotas.	X	X	no
-r *DefaultDataReplicas* See “File system recoverability parameters” on page 36.	X	X	1
-R *MaxDataReplicas* See “File system recoverability parameters” on page 36.	X	This value cannot be changed.	2
-S {**yes** | **no**} to suppress periodic updating of atime values. See “atime values” on page 34.	X	X	no
-t *DriveLetter* See “Windows drive letter” on page 37.	X	X	none
-T *Mountpoint* See “Mountpoint directory” on page 37.	X	X	**/gpfs/***DeviceName*
-V to migrate file system format to the latest level.	NA	X	none
-v {**yes** | **no**} to verify disk usage.	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 39.	X	X	no

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 GPFS clusters. 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**. Do not specify an existing entry in **/dev**.

List of disk descriptors

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 for additional information). 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.

posix Specified for file systems supporting NFS V3 or ones which are not NFS exported. This is the default.

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

Disks for your file system

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

1. Create new disks through the **mmcrnsd** command.
2. Select NSDs no longer in use by another GPFS file system. Issue the **mmlsnsd -F** command to display the available disks.

See "Disk considerations" on page 24.

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, 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 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 contain large files accessed in large reads and writes.
- The 1024 KB or 1 MB block sizes are 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 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 $\times$ 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, **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 represents the time when the file was last accessed. The **-S** parameter controls the updating of the **atime** value. 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** is needed, the application must use the GPFS calls **gpfs_stat()** and **gpfs_fstat()**. When **-S yes** is specified, or the file system is mounted **read-only**, the updating of the **atime** value is suppressed. See “Exceptions to Open Group technical standards” on page 103.

mtime values

mtime represents the time when the file was last modified. The **-E** parameter controls the frequency of updating of the **mtime** value. The default is **-E yes**, which results in 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** value. Note that regardless of the **-E** setting, the GPFS calls **gpfs_stat()** and **gpfs_fstat()** always report the exact mtime. See “Exceptions to Open Group technical standards” on page 103.

Block allocation map

Specifies the block allocation map type. 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 block allocation map can be one of two types:

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 or 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 ACLs are not allowed).

nfs4 Support for NFS V4 ACLs only. Users are not allowed to assign traditional ACLs to any file system objects. This should not be specified unless the file system is going to be exported to NFS V4 clients.

all Allows for the coexistence of POSIX and NFS V4 ACLs within a file system. This should not be specified unless at least one file within the file system is going to be exported to NFS V4 clients.

Strict replication

Strict replication means that data or metadata replication will be 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
Strict replication is always enforced.

The use of strict replication is 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 83 for additional information.

File system recoverability parameters

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

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

Default metadata Replicas

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

Maximum metadata replicas

The maximum number of copies of metadata for all files in the file system 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 failure group that is 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 10, "GPFS architecture," on page 79). Although a larger estimate consumes a bit more memory, insufficient allocation of these data structures can limit node ability to process certain parallel requests efficiently, such as the allotment of disk space to a file. If you cannot 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.

Note: *This value cannot be changed later.*

Maximum number of files

The maximum number of files in a file system may be specified at file system creation by using the **-N** option on the **mmcrfs** command or changed at a later time by using the **-F** option on the **mmchfs** command. This value defaults to the size of the file system at creation divided by 1 MB and cannot exceed the architectural limit of 2,147,483,647.

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

Note:

1. For file systems that will be doing parallel file creates, if the total number of free inodes is not greater than 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 will cause the allocation of too much disk space for control structures.

Windows drive letter

In a Windows environment, you must associate a drive letter with the file system to be mounted. The drive letter can be changed with the **-t** option of the **mmcrfs** and **mmchfs** commands.

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.

Automatic quota activation

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

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

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

1. Soft limit
2. Hard limit
3. Grace period

The soft limits define levels of disk space and files below which the user, group of users, or fileset can safely operate. The hard limits define the maximum disk space and files the user, group of users, or fileset can accumulate. Specify hard and soft limits for disk space 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.

The 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.

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:

- **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 if DMAPI is enabled.

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

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  6 15:10:08 2006
 100 % complete on Mon Mar  6 15:10:11 2006
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
  44 % complete on Mon Mar  6 15:11:32 2006
  90 % complete on Mon Mar  6 15:11:37 2006
```

```
 100 % complete on Mon Mar  6 15:11:38 2006
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  8192           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  posix          File locking semantics in effect
 -k  posix          ACL semantics in effect
 -K  whenpossible   Strict replication enforcement
 -a  1048576        Estimated average file size
 -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
 -F  2998272        Maximum number of inodes
 -V  10.00 (3.2.0.0)  File system version.
 -u  yes            Support for large LUNs?
 -z  no             Is DMAPI enabled?
 -E  yes            Exact mtime mount option
 -S  yes            Suppress atime mount option
 -P  system         Disk storage pools in file system
 -d  hd25n09;hd24n09;hd23n09 Disks in file system
 -A  yes            Automatic mount option
 -o  none           Additional mount options
 -T  /gpfs2         Default mount point
```

Chapter 3. 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. See the GPFS FAQ at 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. Place the GPFS code on your system:
   - For existing systems, see Chapter 7, “Migration, coexistence and compatibility,” on page 59.
   - For new systems:
     - For your Linux nodes, see Chapter 4, “Installing GPFS on Linux nodes,” on page 43.
     - For your AIX nodes, see Chapter 5, “Installing GPFS on AIX nodes,” on page 47.
     - For your Windows nodes, see Chapter 6, “Installing GPFS on Windows nodes,” on page 51.
3. Decide which nodes in your system will be quorum nodes (see “Quorum” on page 15) .
4. Create your GPFS cluster by issuing the **mmcrcluster** command. See “GPFS cluster creation considerations” on page 20.

After your GPFS cluster has been established:

1. Ensure you have configured and tuned your system according to the values suggested in Chapter 8, “Configuring and tuning your system for GPFS,” on page 67.
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 “NSD creation considerations” on page 25.
4. Create new file systems by issuing the **mmcrfs** command. See “File system creation considerations” on page 30.
5. If you need to import file systems that currently belong to some other cluster, see the topic on *Exporting file system definitions between clusters* in the *GPFS: Advanced Administration Guide*.
6. Mount your file systems.
7. As an optional step, you can also create a temporary directory (**/tmp/mmfs**) to collect problem determination data. 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. The **/tmp/mmfs** directory can be a symbolic link to another location if more space can be found there.

   If a problem should occur, GPFS may 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 during the next failure. An alternate path may be specified through the **mmchconfig** command.

Chapter 4. Installing GPFS on Linux nodes

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

Before you begin installation, read Chapter 2, “Planning for GPFS,” on page 13 and the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/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 13 and “Software requirements” on page 13.

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 Linux installation process”
2. “Verifying the level of prerequisite software”
3. “Procedure for installing GPFS on Linux nodes” on page 44
4. “Building your GPFS portability layer” on page 45

Creating a file to ease the Linux installation process

To ease the installation process, it is suggested that you create a file listing all of the nodes in your GPFS cluster using either host names or IP addresses.

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 install 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.

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

- Linux distributions
- RPM levels
- Software recommendations
- Configuration information

Procedure for installing GPFS 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. "Creating the GPFS directory"
3. "Installing the GPFS man pages" on page 45
4. "Installing GPFS over a network" on page 45
5. "Verifying the GPFS installation" on page 45

Accepting the electronic license agreement

The GPFS software license agreement is shipped and viewable electronically. The electronic license agreement must be accepted before software installation can continue. See "Creating the GPFS directory."

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 self-extracting product image, **gpfs_install-3.2***, from the CD-ROM to the new directory (where * is the correct version of the product for your hardware platform and Linux distribution).

   The image contains:

   - The GPFS product installation images
   - 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

3. Verify that the self-extracting program has executable permissions.
4. 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 will be extracted to the target directory **/usr/lpp/mmfs/3.2*** .
   b. The license agreement files on the media can be viewed in graphics mode or text-only mode.

      To view the files in graphics mode, invoke **gpfs_install-3.2***. To view the files in text-only mode, use the **--text-only** option.
   c. Use the **--silent** option to accept the license agreements.
   d. Use the **--help** option to obtain usage information from the self-extracting archive.

   ```
   gpfs_install-3.2.* --silent
   ```

Upon license agreement acceptance, the GPFS product installation images will reside in the extraction target directory. Copy these images to the **/tmp/gpfslpp** directory:

1. gpfs.base-3.2*.rpm
2. gpfs.gpl-3.2*.noarch.rpm
3. gpfs.msg.en_US-3.2*.noarch.rpm
4. gpfs.docs-3.2*.noarch.rpm

The License agreements will remain available in the extraction target directory under the **license** subdirectory for future access. The license files are written using operating system-specific code pages. Accordingly, you may view the license in English and the local language configured on your machine. The other languages are not guaranteed to be viewable.

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 will be located at **/usr/share/man/**.

Note: The **gpfs.docs** RPM need not be installed on all nodes if man pages are not desired or local file 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:

   ```
   cat /tmp/gpfs.allnodes | xargs -i rsh {} 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:

   ```
   cat /tmp/gpfs.allnodes | xargs -i rsh {} mount localNode:/tmp/gpfslpp /tmp/gpfslpp
   ```

   Otherwise, issue:

   ```
   cat /tmp/gpfs.allnodes | xargs -i rcp /tmp/gpfslpp/gpfs*.rpm {}:/tmp/gpfslpp
   ```
4. Install GPFS on each node:

   ```
   cat /tmp/gpfs.allnodes | xargs -i rsh {} rpm -Uvh /tmp/gpfslpp/gpfs*.rpm
   ```

Verifying the GPFS installation

Verify the installation of GPFS file sets on each system node to check that the software has been successfully installed:

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

The system should return output similar to:

```
gpfs.docs-3.2.1-0
gpfs.base-3.2.1-0
gpfs.msg.en_US-3.2.1-0
gpfs.gpl-3.2.1-0
```

Building your GPFS portability layer

Before starting GPFS, you must build the GPFS portability layer, which is a set of binaries that need to be built locally from source code to match your Linux kernel and configuration.

1. Before building the portability layer check for:
   - Updates to the portability layer at http://www14.software.ibm.com/webapp/set2/sas/f/gpfs/home.html
   - The latest kernel level support in the GPFS FAQ at 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 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 directions in **/usr/lpp/mmfs/src/README**
   - Using the Autoconfig tool.

Using the automatic configuration tool to build GPFS portability layer

To help you build the portability layer, GPFS provides an automatic configuration tool. This tool, named **configure** is a Perl script residing in **/usr/lpp/mmfs/src/config/configure** on installed GPFS systems. When you run the **configure** tool, it automatically examines the system and generates a working configuration file. GPFS will then use the configuration file as it creates the portability layer.

Note: You must manually invoke the **configure** tool.

While it is running, the **configure** tool gathers system-specific parameters and combines them with a template file to produce a configuration file for the build. The **makefile** invocation for the **configure** tool is **make Autoconfig** which must be issued from the top level **$SHARKCLONEROOT** directory. The configuration file is then invoked by issuing **make World** in the usual manner. The invoker has the option to specify input parameters as well as pairs of attributes and values to change the configuration attributes. Use **configure -help** to see a list of options and attributes.

Note:

1. The environment variable **SHARKCLONEROOT** is the root of the GPFS source
2. If **SHARKCLONEROOT** is not set, **configure** uses **/usr/lpp/mmfs/src**

This example shows the commands to create the configuration file and then invoke it to make the portability layer:

```
cd /usr/lpp/mmfs/src
export SHARKCLONEROOT=/usr/lpp/mmfs/src
make Autoconfig
#Can check /usr/lpp/mmfs/src/config/site.mcr
make World
make InstallImages
echo $?
```

Chapter 5. Installing GPFS on AIX nodes

There are three steps to installing GPFS on AIX 5L™ nodes. The information in this topic will point you to the detailed steps.

Before you begin installation, read Chapter 2, "Planning for GPFS," on page 13 and the GPFS FAQs at publib.boulder.ibm.com/infocenter/clresctr/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 13 and "Software requirements" on page 13.

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 48

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, you must 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 at: 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:

```
5.3.0.10
```

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 49
6. "Reconciling existing GPFS files" on page 49
7. "Verifying the GPFS installation" on page 49

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                  3.2.1.5  COMMITTED  GPFS File Manager
  gpfs.msg.en_US             3.2.1.5  COMMITTED  GPFS Server Messages - U.S.
                                                 English
```

```
Path: /etc/objrepos
  gpfs.base                 3.2.1.5  COMMITTED  GPFS File Manager

Path: /usr/share/lib/objrepos
  gpfs.docs.data            3.2.1.5  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 6. 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.

Do not install GPFS if you do not have the prerequisites listed in “Hardware requirements” on page 13 and “Software requirements” on page 13.

Before you begin installation, read the following:

- Chapter 2, “Planning for GPFS,” on page 13
- The GPFS FAQ at: 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

The installation process includes:

1. “Installing GPFS prerequisites” on page 55
2. “Procedure for installing GPFS on Windows nodes” on page 58
3. “Configuring Windows” on page 56

To install GPFS on Windows, first configure your Windows systems as described in “Installing GPFS prerequisites” on page 55. This includes installing SUA, creating a GPFS administrative account, and installing SSH. GPFS installation will be simple once the prerequisites are completed.

Note: Throughout the GPFS documentation, there are directory and file names given as UNIX-style paths such as **/tmp/mmfs** and **/var/mmfs/gen/mmsdrfs**. These files exist on Windows under the **%SystemRoot%\SUA** directory. This means that a file referenced as **/var/mmfs/gen/mmsdrfs** in the GPFS documentation will have a name like **C:\Windows\SUA\var\mmfs\gen\mmsdrfs** on Windows nodes.

GPFS for Windows overview

GPFS for Windows Multiplatform V3.2.1 supports the Windows Server 2003 R2 operating system running on 64-bit architectures (AMD x64 / EM64T) in an AIX or Linux GPFS cluster.

GPFS for Windows participates in a new or existing GPFS V3.2 cluster in conjunction with AIX and Linux (32- or 64-bit) systems. Support includes:

- Client access to GPFS V3.2 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 one of the key features of GPFS for Windows Multiplatform. 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 Multiplatform provides the same core services to parallel and serial applications as are available on AIX and Linux. GPFS gives parallel applications simultaneous access to the same files, or different files, from any node that has GPFS mounted, while managing a high level of control over all file system operations. System administrators and users have a consistent command interface on AIX, Linux, and Windows. With few exceptions, the commands supported on Windows are identical to those on other GPFS platforms. See "GPFS limitations on Windows" for a list of commands that Windows clients do not support.

GPFS limitations on Windows

GPFS for Windows does not fully support all the GPFS features 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:

- Windows nodes do not support direct access to disks. Windows nodes in a GPFS cluster can only access storage as a network shared disk (NSD) client. This means that any cluster that includes Windows requires at least one AIX or Linux node operating as a NSD server.
- 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 created with a GPFS version older than V3.2.
- File systems cannot be shared with other GPFS clusters. Clusters that contain Windows nodes cannot share its file systems with other GPFS clusters. Similarly, these clusters cannot remote-mount file systems from other clusters. (See *Accessing GPFS file systems from other GPFS clusters* in the *GPFS: Advanced Administration Guide*.)
- The cluster cannot use OpenSSL. Clusters that contain Windows nodes cannot use OpenSSL for connection authentication or data encryption in GPFS.
- 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:

- Windows nodes cannot be assigned file system manager or quorum roles.
- The Tivoli® Storage Manager (TSM) Backup Archive client for Windows does not support GPFS file systems. TSM backup and archiving operations are supported on AIX and Linux nodes in a cluster that contains Windows. For information on TSM backup archive client support for GPFS, see:
  - The GPFS FAQ at http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html
  - IBM Tivoli Storage Manager Support at http://www.ibm.com/software/sysmgmt/products/support/IBMTivoliStorageManager.html
- The following GPFS commands are not supported on Windows:
  - **mmapplypolicy**
  - **mmbackup**
  - **mmcheckquota**
  - **mmdefedquota**
  - **mmdelacl**
  - **mmeditacl**
  - **mmedquota**
  - **mmgetacl**
  - **mmlsquota**
  - **mmpmon**

- **mmputacl**
- **mmrepquota**
- **mmrestripefile**

- 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 compute 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 name considerations

File names created on UNIX-based GPFS nodes using characters that are not valid for the Windows file systems (such as colons, slashes, back slashes, asterisks, question marks, double quotation marks, less than, greater than, and pipe characters) are transformed into valid short names. Windows applications can use the short name to gain access to files. GPFS generates unique short names using an internal algorithm. You can view these short names by issuing **dir /x** in a command prompt.

Table 5 shows an example:

Table 5. Generating short names for Windows

UNIX	Windows
foo+bar.foobar	FO~23Q_Z.foo
foolbar.-bar	FO~TD}C5._ba
flbar.-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 means that GPFS does not provide efficient support for the indexing services accessible through the Windows **Search for files or folders** function.

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 a Windows domain to support a GPFS cluster, and details on setting up individual Windows systems prior to installing GPFS.

Restriction: GPFS runs *only* on Windows Server 2003 R2 x64 with SP2 *or higher*.

Perform the following steps:

1. Set up the Windows domain (see "Setting up the Windows domain").
2. Create an administrative account (see "Creating the GPFS administrative account" on page 56).
3. Configure network settings (see "Configuring Windows" on page 56).
4. Install SUA (see "Installing the Subsystem for UNIX-based Applications" on page 57).
5. Install SUA Hotfix updates (see "Downloading and installing SUA hotfix updates" on page 57).
6. Install and configure remote shell and copy (see "Installing and configuring OpenSSH" on page 57).

See the GPFS FAQ at 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

To perform these operations, you must be a member of the Administrators group on the local computer or a member of the Domain Admins group. To perform GPFS administrative operations, you should login as the special root user account described in "Creating the GPFS administrative account" on page 56.

Setting up the Windows domain

GPFS for Windows runs on systems that are part of a Windows domain. It relies on Active Directory Domain Services to provide users with a consistent identity between the Windows nodes in a cluster. GPFS can also exploit a Windows Server feature called Identity Management for UNIX (IMU) to provide consistent identities between all nodes in a cluster.

GPFS expects that Windows nodes in a cluster are members of the same Windows 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.

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 2003R2 that can be installed on domain controllers. GPFS does not require IMU.

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

Tip: Your Windows domain might not have its own Domain Name System (DNS) subdomain. If this is the case, explicitly set each node's primary DNS suffix and disable the option to change primary DNS suffix when domain membership changes. For details on how to make this change, search for *Change the DNS suffix of your computer* in your system's Help and Support Center.

Creating the GPFS administrative account

GPFS uses an administrative account in the Windows 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. Specify in root's profile a **Home** folder in **Local path** that does not include spaces in the path name, such as **C:\Users\root**.

Step 3 avoids problems in the SUA environment (described in "Installing the Subsystem for UNIX-based Applications" on page 57) that occur because the default value for a user's **HOME** directory on Windows Server 2003 is a path like **C:\Documents and Settings\root**, which contains spaces. The **HOME** directory specified in root's profile should get created the first time root logs on a system. If root's **HOME** is created by other means, be sure root is the directory's owner.

For more information, see the FAQ at http://www.interopsystems.com/community/faqs.aspx

Configuring Windows

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

GPFS supports Microsoft Windows Server 2003 R2 for x64-Based Systems, and requires Service Pack 2 (SP2). Both the Standard and Enterprise editions of this operating system are supported.

All Windows systems that will be added to the same GPFS cluster should be members of the same Windows domain.

GPFS requires two optional system components included with Windows Server 2003 R2:

- Microsoft .NET Framework 2.0
- Subsystem for UNIX-based Applications (see "Installing the Subsystem for UNIX-based Applications" on page 57)

To install these optional components, open **Add or Remove Programs** in the Control Panel, and then click **Add/Remove Windows Components**.

GPFS also 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 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

The Subsystem for UNIX-based Applications (SUA) is a POSIX subsystem included with Windows Server 2003 R2. GPFS uses this component to support many of its programs and administrative scripts. System administrators have the option of using either a SUA shell such as **ksh** or the standard Windows Command Prompt to run GPFS commands.

The SUA environment is composed of two parts:

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

Both parts must be installed before installing GPFS. The Subsystem provides runtime support for POSIX applications and is a component included with the Windows operating system. The Utilities 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**.

Note: Install SUA on each Windows node *after* adding it to the Windows domain (otherwise, uninstall the SUA SDK and SUA, and then reinstall them).

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

1. Add the POSIX Subsystem (an optional component on Server 2003 R2) by clicking **Control Panel → Add or Remove Programs → Add/Remove Windows Components**.
2. While this component is being installed, you are prompted to download the Utilities and SDK. Click **yes** if you have not already downloaded this package.
3. Start the Utilities installation. The default options are not sufficient for GPFS, so select **Entire feature** at the top level to include all elements. On the Security Settings panel, check **Enable setuid behavior for SUA programs** and check **Change the default behavior to case sensitive**.

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 athttp://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 clicking through the **Add or Remove Programs** control panel. Select the **Show updates** option at the top.

Installing and configuring OpenSSH

GPFS uses the SUA Community version of OpenSSH to support its administrative functions. Microsoft does not provide SSH support with SUA, and the remote shell (**rsh**) service included with SUA has limitations that make it unsuitable for GPFS. Interop Systems Inc. hosts the SUA Community Web site (http://www.interopsystems.com/community/), which includes a forum and other helpful resources related to SUA and Windows/UNIX interoperability.

The steps below outline a procedure for installing OpenSSH. This information could change at any time. Refer to the SUA Community Web site (http://www.suacommunity.com/forum/default.aspx) for the current and complete installation instructions. Although Interop Systems provides SUA Add-on Bundles that include OpenSSH and many other packages, IBM recommends installing only the SUA Community packages required in your environment.

1. Register with the site (http://www.suacommunity.com/forum/register.aspx).

2. Download the Boostrap Installer (3.5) (**ftp://ftp.interopsystems.com/pkgs/bootstrap/pkg-current-bootstrap35.exe**) and install it on your Windows nodes.
3. From a SUA shell, run **pkg_update -L openssh**.

Once OpenSSH is installed and configured, verify that the user **root** can issue **ssh** and **scp** commands between nodes in the GPFS cluster without being prompted for a password. The **ssh** and **scp** commands should be verified as working using just the host name and the full DNS name of every node in the cluster, including the local node. For more information about **ssh**, see the *Troubleshooting Windows* topic in the *GPFS: Problem Determination Guide*.

Note: You must ensure that nodes in the GPFS cluster can issue remote shell (**ssh**) and copy (**scp**) commands without the use of a password as user **root**.

Procedure for installing GPFS on Windows nodes

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

For more information, see “Installing GPFS prerequisites” on page 55.

To install GPFS, follow these steps:

1. Run the setup program, **gpfs-3.2.1.5-WindowsServer2003R2-x64.msi**, from the product media and accept the license.

   For more information, refer to Chapter 3, “Steps to establishing and starting your GPFS cluster,” on page 41.
2. Download and install the latest service level of GPFS from the GPFS support site at http://www14.software.ibm.com/webapp/set2/sas/f/gpfs/home.html

Chapter 7. Migration, coexistence and compatibility

For the latest information on migration, coexistence, and compatibility, see the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html

To migrate to GPFS 3.2, you must first consider whether you are migrating from GPFS 3.1 or an earlier release of GPFS, and then consider coexistence and compatibility issues.

GPFS migration consists of these topics:

Migrating to GPFS 3.2 from GPFS 3.1

GPFS 3.2 supports node-at-a-time migration if the previous nodes in the cluster are running GPFS 3.1.
| GPFS 3.1 nodes can coexist and interoperate with nodes running GPFS 3.2. However, some new
| functions (such as creating a GPFS file system to be mountable on Microsoft Windows), that depend on
| format changes will not be available until all nodes have been migrated.

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

1. Stop all user activity in the file systems on the designated node.
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 new code on each node in the cluster as described in Chapter 4, "Installing GPFS on Linux nodes," on page 43 or Chapter 5, "Installing GPFS on AIX nodes," on page 47.
6. Start GPFS on the designated node in the new cluster, for example, issue:
   ```
   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" on page 62.

Migrating to GPFS 3.2 from GPFS 2.3

You can migrate to GPFS 3.2 from GPFS 2.3 by following the steps in this procedure.

To migrate a cluster to GPFS 3.2 from GPFS 2.3, 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. 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 new code on each node in the cluster as described in Chapter 4, “Installing GPFS on Linux nodes,” on page 43 or Chapter 5, “Installing GPFS on AIX nodes,” on page 47.
6. Start GPFS on all nodes in the new 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” on page 62.

Migrating to GPFS 3.2 from GPFS 2.2 or earlier releases of GPFS

You can migrate to GPFS 3.2 from GPFS 2.2 or earlier releases by following the steps in this procedure.

In release 2.2 and earlier, GPFS could be configured in a number of cluster types: **hacmp**, **lc**, **rpd**, and **sp**. The different cluster types supported different disk types such as virtual shared disks, AIX logical volumes, and NSDs. In addition, one could divide a GPFS cluster into a number of independent node sets, which determined the scope of the nodes on which a given file system could be mounted.

Starting with GPFS 2.3, many of the following concepts have been eliminated and streamlined. For example:

- The only (implied) GPFS cluster type is **lc**.
- The concept of node sets is eliminated.
- All nodes in the GPFS cluster are now automatically members of the one and only node set that you can have in a GPFS cluster.
- The dependence on RSCT has been removed.
- The only disk type that you can have is NSD.

  NSDs can be created out of any of the existing disk types, preserving file systems created prior to GPFS 2.3.

All of these changes, plus some of the new features that are introduced with GPFS 2.3, require that you rebuild your existing cluster. For a complete list of new functions and changes, see the *Summary of changes*. For an overview and description of GPFS clusters and NSD, see Chapter 1, “Introducing General Parallel File System,” on page 1.

If you currently use more than one node set, you need to decide whether you want to combine the node sets into one GPFS cluster or create more than one GPFS cluster to correspond to each of your node sets. The migration procedure assumes that you are going to create a single GPFS cluster and move all of your file systems into it. You can always create new clusters later and move your file systems around using the **mmexportfs** and **mmimportfs** commands.

Note:

1. Before you proceed, it is recommended that you read the notes that are specific to your existing cluster environment.
2. **GPFS cluster type sp:** The **mmcrcluster** command is not available on this cluster type. If you have an **sp** cluster type, use the **mmlsnode -a** command to obtain the names of your node sets and the nodes that belong to them.

To determine your cluster type, issue the **mmlscluster** command, which displays output similar to:

```
GPFS cluster information
========================
  GPFS cluster type:         rpd
  GPFS cluster id:           gpfs0103263150103
  RSCT peer domain name:     rpd12
  Remote shell command:      /usr/bin/rsh
  Remote file copy command:  /usr/bin/rcp

GPFS cluster data repository servers:
-------------------------------------
  Primary server:    k145n43.kgn.ibm.com
  Secondary server:  k145n44.kgn.ibm.com

Nodes in nodeset set1:
----------------------
    1   k145n29     9.114.133.29  k145n29.kgn.ibm.com
    2   k145n30     9.114.133.30  k145n30.kgn.ibm.com
    3   k145n31     9.114.133.31  k145n31.kgn.ibm.com
    4   k145n32     9.114.133.32  k145n32.kgn.ibm.com

Nodes in nodeset set2:
----------------------

    5   k145n33     9.114.133.33   k145n33.kgn.ibm.com
    6   k145n34     9.114.133.34   k145n34.kgn.ibm.com
    7   k145n35     9.114.133.35   k145n35.kgn.ibm.com
    8   k145n36     9.114.133.36   k145n36.kgn.ibm.com
    9   k145n37     9.114.133.37   k145n37.kgn.ibm.com
   10   k145n38     9.114.133.38   k145n38.kgn.ibm.com
   11   k145n39     9.114.133.39   k145n39.kgn.ibm.com
   12   k145n40     9.114.133.40   k145n40.kgn.ibm.com
   13   k145n41     9.114.133.41   k145n41.kgn.ibm.com
   14   k145n42     9.114.133.42   k145n42.kgn.ibm.com
   15   k145n43     9.114.133.43   k145n43.kgn.ibm.com
   16   k145n44     9.114.133.44   k145n44.kgn.ibm.com
   17   k145n45     9.114.133.45   k145n45.kgn.ibm.com
   18   k145n46     9.114.133.46   k145n46.kgn.ibm.com

Cluster nodes that are not assigned to a nodeset:
-------------------------------------------------
   19  k145n47      9.114.133.47   k145n47.kgn.ibm.com
```

- If your current cluster type is **lc**:

  Because your disks are already defined as NSDs, there are no additional considerations. Some or all of your nodes are members of an RSCT peer domain. Because GPFS no longer depends on RSCT, whether you continue to keep the nodes in the domain from now on should be dictated by other non-GPFS considerations you may have. GPFS will not be affected one way or another.

- If your current cluster type is **hacmp** or **rpd**:

  Your current GPFS disks are either virtual shared disks or AIX logical volumes. They will continue to be the same, but GPFS automatically converts them into NSDs as part of the **mmimportfs** command processing (see step 13 on page 62). During the import process, even though your disks are converted to NSDs, the names of your disks are preserved. For example, you may have NSDs with the names **gpfs5vsd** or **gpfs7lv**. This does not affect the performance of your file system in any way.

  Although GPFS does not depend on RSCT any more, if your current disks are virtual shared disks, you will have to maintain the node membership in the RSCT peer domain because of the requirements of the IBM virtual shared disk subsystem.

- If your current cluster type is **sp**:

  Your current GPFS disks are virtual shared disks. They will continue to be the same, but GPFS will automatically convert them into NSDs as part of the **mmimportfs** command processing (see step 13 on page 62). During the import process, even though your disks are converted to NSDs, the names of your disks are preserved. For example, you may have an NSD with the name **gpfs5vsd**. This does not affect the performance of your file system in any way.

To migrate your GPFS cluster to GPFS 3.2, follow these steps:

1. Ensure that all disks in all GPFS file systems to be migrated are in working order by issuing the **mmlsdisk** command. Verify that the disk status is **ready** and availability is **up**. If not, correct any problems and reissue the **mmlsdisk** command before continuing.
2. Stop all user activity in the file systems.
3. Follow any local administrative backup procedures to ensure protection of your file system data in the event of a failure.
4. Cleanly unmount all mounted GPFS file systems. Do not use force unmount.
5. Shut down the GPFS daemon on all nodes in the cluster:

   ```
   mmshutdown -a
   ```
6. Export the GPFS file systems by issuing the **mmexportfs** command:

   ```
   mmexportfs all -o exportDataFile
   ```

   This command creates the configuration output file *exportDataFile*, which contains all exported configuration data. Retain this file because it is required when issuing the **mmimportfs** command to import your file systems into the new cluster or in the event that you decide to go back to the previous release.
7. Delete all existing nodes. For each node set in the cluster, where *nodesetId* is the name of the node set, issue:

   ```
   mmdelnode -a -C nodesetId
   ```
8. Delete the existing cluster by issuing:

   ```
   mmdelcluster -a
   ```

   Note: This step does not apply to GPFS cluster type **sp**.
9. Ensure that all disks from the old GPFS cluster are properly connected and are online and available to the appropriate nodes of the new GPFS cluster.
10. Install the new level of GPFS on the affected nodes:
    - For Linux nodes, see Chapter 4, “Installing GPFS on Linux nodes,” on page 43
    - For AIX nodes, see Chapter 5, “Installing GPFS on AIX nodes,” on page 47
11. Decide which nodes in your system will be quorum nodes (see “Quorum” on page 15).
12. Create a new GPFS cluster across all desired nodes by issuing the **mmcrcluster** command.
13. To complete the movement of your file systems to the new cluster, using the configuration file created in step 6, issue:

    ```
    mmimportfs all -i exportDataFile
    ```
14. Start GPFS on all nodes in the new cluster:

    ```
    mmstartup -a
    ```
15. Mount the file systems if this is not done automatically when the GPFS daemon starts.
16. Proceed to “Completing the migration to a new level of GPFS.”

Completing the migration to a new level of GPFS

You should operate GPFS with the new level of code 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.

Note: If you need to revert to the previous level, see “Reverting to the previous level of GPFS” on page 64 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 code:

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

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

   - If you issue **mmchfs -V compat**, it enables only backward-compatible format changes. Nodes in remote clusters that were able to mount the file system before will continue to be able to do so.
   - If you issue **mmchfs -V full**, it enables all new functions that require different on-disk data structures. Nodes in remote clusters running an older GPFS version will no longer be able to mount the file system. If there are any nodes running an older GPFS version that have the file system mounted at the time this command is issued, the **mmchfs** command will fail.

   To enable backward-compatible format changes, issue:

   ```
   mmchfs filesystem -V compat
   ```

   To migrate all file systems to the latest metadata format changes, issue:

   ```
   mmchfs filesystem -V full
   ```

   where: *filesystem* is the name of the file system.
3. If you were using single-node quorum in versions prior to GPFS 2.3, you must transition to the new node quorum with tiebreaker disks. See "Quorum" on page 15.

| **Note:** Only file systems at GPFS 3.2.1.5*x* are mountable on Windows nodes.

Additional considerations when migrating GPFS 2.3 and earlier file systems

The concept of storage pools and filesets does not exist in file systems that are at version level 8.00 (GPFS 2.3) or earlier. When such a file system is migrated to the latest GPFS level, the migration process automatically creates a ″root″ fileset and ″system″ storage pool.

Root fileset

During migration, all existing files and directories are assigned to the root fileset. After migration, you can:

- Create new filesets with **mmcrfileset**
- Link filesets into the name space with **mmlinkfileset**
- Copy data into the filesets using ordinary file system tools such as **cp** or **tar**

Note: The migration process will not automatically enable fileset quotas in quota enabled file systems. You must do this explicitly using the **mmchfs -Q yes** command. That command will automatically create the **fileset.quota** file in the root directory of the file system

System storage pool

During migration, all of the disks are assigned to the **system** storage pool. When you add new disks, you can assign them to other storage pools. Adding new disks automatically creates the storage pool named in the disk descriptors. To move a disk already belonging to the file system to a new storage pool use **mmdeldisk** followed by **mmadddisk**.

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.

You can revert back to either GPFS 3.1 or GPFS 2.3. 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 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 code 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 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 code 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. Import the file system information using the **mmimportfs** command. Specify the file created by the **mmexportfs** command from Step 4 on page 64 above:

    ```
    mmimportfs all -i exportDataFile
    ```

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

    ```
    mmstartup -a
    ```

14. 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.

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 3.2.1.

Considerations for IBM Tivoli Storage Manager for Space Management

Migrating to GPFS V3.2 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 that some nodes in a cluster are at GPFS V3.2, and some nodes in the cluster are at a lower level. During that time, the DMAPI file handle size can be 16 bytes. Until all nodes in the cluster have been updated to GPFS V3.2, and the DMAPI file handle size is changed to 32 bytes, IBM Tivoli Storage Manager for Space Management *must* be disabled.

After all nodes in the cluster are upgraded to GPFS 3.2 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 3.2 and GPFS 3.1 nodes. This function is intended for short-term use. You will not get the full functions of GPFS 3.2 until all nodes are using GPFS 3.2.
2. Interoperability between maintenance levels will be specified on a per-maintenance-level basis. See the GPFS FAQ at: 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-3.2-1-1.i386.update.rpm**.
4. For the latest service information, see the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html

To download fixes for GPFS go to: 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 4, "Installing GPFS on Linux nodes," on page 43.
- For AIX nodes, see Chapter 5, "Installing GPFS on AIX nodes," on page 47.
- For Windows nodes, see Chapter 6, "Installing GPFS on Windows nodes," on page 51.

Chapter 8. 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 20 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 at publib.boulder.ibm.com/infocenter/clresctr/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 70
3. “AIX configuration and tuning considerations” on page 73
4. “Configuring Windows” on page 56

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*
- *Large system considerations* for information on 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 at publib.boulder.ibm.com/infocenter/clresctr/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 68
4. “GPFS I/O” on page 69
5. “Access patterns” on page 70
6. “Aggregate network interfaces” on page 70
7. “Swap space” on page 70

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**. If you have designated the use of a different remote communication program, you must make certain that:
  - You have granted proper authorization to all nodes in the GPFS cluster.
  - The nodes in the GPFS cluster can communicate without the use of a password and without any extraneous messages.

  Note: If you are using **rcp** and **rsh** commands for remote communication, the root user must have a properly configured **.rhosts** file in the home directory on each node in the GPFS cluster.

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 100 MB:

```
mmchconfig pagepool=100M
```

maxFilesToCache

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

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

maxFilesToCache × **2.5 KB**

Valid values of **maxFilesToCache** range from 1 to 100,000. 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 × 176 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** × 2.5 KB), and the memory for the stat cache (**maxStatCache** × 176 bytes) together. The combined amount of memory to hold inodes, control data structures, and the stat cache is limited to 50% of the physical memory 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 83.

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. "SUSE LINUX considerations"
3. "GPFS helper threads"
4. "Communications I/O"
5. "Disk I/O" on page 72

updatedb considerations

On some Linux distributions, for example, Red Hat EL 3.0, 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.

SUSE LINUX considerations

On the SUSE LINUX ES 9 distribution, it is recommended 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.

Also, please see the GPFS Redpapers:

- *GPFS Sequential Input/Output Performance on IBM pSeries® 690* at www.redbooks.ibm.com/redpapers/pdfs/redp3945.pdf
- *Native GPFS Benchmarks in an Integrated p690/AIX and x335/Linux Environment* at 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=18
mmchconfig worker1Threads=24
```

Communications I/O

To optimize the performance of GPFS and your network, it is suggested you:

- 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.

If GPFS is configured to operate over Myrinet, to enable the Jumbo Frames for Myrinet IP driver build the GM driver with the **--enable-new-features**.

- 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. Suggested settings for the IBM TotalStorage® DS4500 include:
   - read cache = enabled
   - read ahead multiplier = 0
   - write cache = disabled
   - write cache mirroring = disabled
   - cache block size = 16K

   Note: Other storage server brands may have similar settings.
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

Configuration and tuning considerations for the AIX nodes in your system include: I/O for communications and GPFS disks, the switch pool, and the possible use of IBM Virtual Shared Disks or Oracle.

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 information about configuration and tuning considerations for AIX nodes, see the following topics:

1. "Communications I/O"
2. "Disk I/O"
3. "Switch pool" on page 74
4. "eServer High Performance Switch" on page 74
5. "IBM Virtual Shared Disk" on page 74
6. "GPFS use with Oracle" on page 75

Communications I/O

There are two considerations for communications within your cluster:

1. For a cluster utilizing the HPS:
   - Configure GPFS to communicate and pass administrative traffic over the LAN, not the switch.
   - Attach and configure AIX nodes designated as virtual shared disk servers, to use the switch.
2. The **ipqmaxlen** network option should be considered when configuring for GPFS. The **ipqmaxlen** parameter controls the number of incoming packets that can exist on the IP interrupt queue. Since both GPFS and IBM Virtual Shared Disk use IP, the default of 100 is often insufficient. The suggested setting is 512. To set the value to 512 after the next reboot, issue the command:

   ```
   no -r -o ipqmaxlen=512
   ```

   This value will persist throughout subsequent reboots. For detailed information on the **ipqmaxlen** parameter, see the *AIX 5L Performance Management Guide* for AIX V5.3 at http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp.

Disk I/O

The disk I/O option to consider when configuring GPFS and using SSA RAID:

max_coalesce

The **max_coalesce** parameter of the SSA RAID device driver allows the device driver to coalesce requests which have been broken up to satisfy LVM requirements. This parameter can be critical when using RAID and is required for effective performance of RAID **write**s. The suggested setting is 0x40000 for RAID-5**4+P** configurations.

- To view:

  ```
  lsattr -E -l hdiskX -a max_coalesce
  ```

- To set:

  ```
  chdev -l hdiskX -a max_coalesce=0x40000
  ```

For further information on the **max_coalesce** parameter see the *AIX 5L Technical Reference: Kernel and Subsystems, Volume 2* for AIX V5.3 at http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp.

Switch pool

If your hardware configuration consists of the eServer HPS, there are two communication subsystem parameters which must be considered when tuning for GPFS. They are **rpoolsize** and **spoolsize**. For optimal system performance, you must allocate sufficient memory for use by these switch pools.

- Switch receive pool, **rpoolsize**, is the pool of memory which is allocated for buffers being received from the switch. Shortages of these buffers will result in dropped packets and retries at higher protocol levels.
- Switch send pool, **spoolsize**, is the pool of memory which is used for all switch output. Shortages of these buffers will result in a delay of I/O operations. On virtual shared disk servers, shortages of these buffers may result in idle disks because buddy buffers can not be freed.

For the HPS, the default value for **rpoolsize** and **spoolsize** is 67108864. The maximum allowed value for both parameters is 134217728. When setting the **rpoolsize** and **spoolsize** parameters for a individual node, the value should scale with the number of HPS links.

- For each link on a node you need to allocate 16 Megabytes of **rpoolsize** and **spoolsize**.
- For a 2 link system, the **rpoolsize** and **spoolsize** values should be 33554432.
- For an 8 link node the values should be set to 134217728.
- For other configurations the value should be proportional.

The suggested setting on virtual shared disk servers is the default value. Under some loads, lesser values may be sufficient on application nodes without disks attached to them.

- To verify the value of spoolsize and rpoolsize:

  ```
  lsattr -E -l sni0
  ```

- To change the value:

  ```
  chgsni sn0 -a rpoolsize=33554432 -a spoolsize 33554432
  ```

For a complete listing of the required AIX file sets and for information on network tuning and Technical Large Page Support configuration, refer to the *Switch Network Interface for eServer High Performance Switch Guide and Reference*, (SC23-4869):

- For AIX V5.3, go to http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp.

eServer High Performance Switch

If your system consists of the eServer High Performance Switch, it is suggested that you configure GPFS over the **ml0** IP network interface.

IBM Virtual Shared Disk

There are three IBM Virtual Shared Disk configuration settings for the efficient operation of GPFS:

1. The buddy buffer is pinned kernel memory. The virtual shared disk server node uses the buddy buffer to temporarily store data for I/O operations originating at a client node. The data in a buddy buffer is purged immediately after the I/O operation completes. The values associated with the buddy buffer are:
   - Minimum buddy buffer size allocated to a single request
   - Maximum buddy buffer size allocated to a single request
   - Total number of maximum-size buddy buffers that the system will attempt to dynamically allocate

   Suggested values (unless your system is memory-constrained and you want to restrict the amount of buddy buffer space) are:
   - Minimum buddy buffer size: 4096 (4 KB)
   - Maximum buddy buffer size: 262144 (256 KB)

If your application uses the **fastpath** option of asynchronous I/O, the maximum buddy buffer size must be greater than or equal to 128 KB. Otherwise, you will receive **EMSGSIZE Message too long** errors.

- Total number of maximum-size buffers:
  - 2000 for a virtual shared disk server node
  - One for a virtual shared disk client-only node

You can either set these values using the **vsdnode** command, or update the values using the **updatevsdnode** command.

When the device driver is configured, the total buddy buffer space is not pinned; instead, approximately one-quarter of the total space requested is pinned when the device driver is configured. This initial amount of spaced pinned is limited to a maximum of 64 MB for a 32-bit kernel, or 128 MB for a 64-bit kernel. After configuration, the device driver attempts to dynamically expand and contract additional buddy buffer space up to the maximum specified, or until AIX can no longer satisfy the memory request. If a buddy buffer cannot be obtained, then the request is queued at the virtual shared disk server until a buddy buffer is available.

2. The **IP_max_msg_size** parameter controls the size of the packets that the IBM Virtual Shared Disk will send between the client and the server. Larger message sizes will result in fewer packets to transfer the same amount of data. The suggested setting is 61440 on all nodes.

Note: For virtual shared disks in a system utilizing the HPS, see the *Reliable Scalable Cluster Technology: Managing Shared Disks* manual for information on buddy buffers. This document is located at http://publib.boulder.ibm.com/infocenter/clresctr/topic/com.ibm.cluster.rsct.doc/rsctbooks.html.

GPFS use with Oracle

When utilizing GPFS with Oracle, configuration and tuning considerations include:

- When setting up your LUNS it is important to:
  1. Create the logical volumes such that they map one to one with a volume group and a volume group be one to one with a LUN which is a single RAID device.
  2. Not stripe logical volumes across multiple RAIDs for I/O parallelism when using raw logical volumes because:
     - GPFS puts each block of a file on different LUNs spread across all LUNs
     - Logical volume striping makes removing a bad RAID more difficult
- 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:
  - On a 64-bit AIX kernel, the 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.
  - On a 32-bit kernel, the setting can be as large as 162.

    The GPFS prefetch threads must be adjusted accordingly through the **mmchconfig - prefetchThreads** command as the sum of those two classes of threads must be 164 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 9. 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. Click **Add or Remove Programs** in the Control Panel.
     b. Click **IBM General Parallel File System**.
     c. Click **Remove**, which will step you through the uninstall process. A reboot will be required.
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 10. 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 15 for more information.
The cluster manager performs the following tasks:

- Monitors disk leases
- Detects failures and drives 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 *Large system considerations* in *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 *Large system considerations* in *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 same 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

   Note: 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.

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: metadata, quota files, GPFS recovery logs, and user data.

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

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

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

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

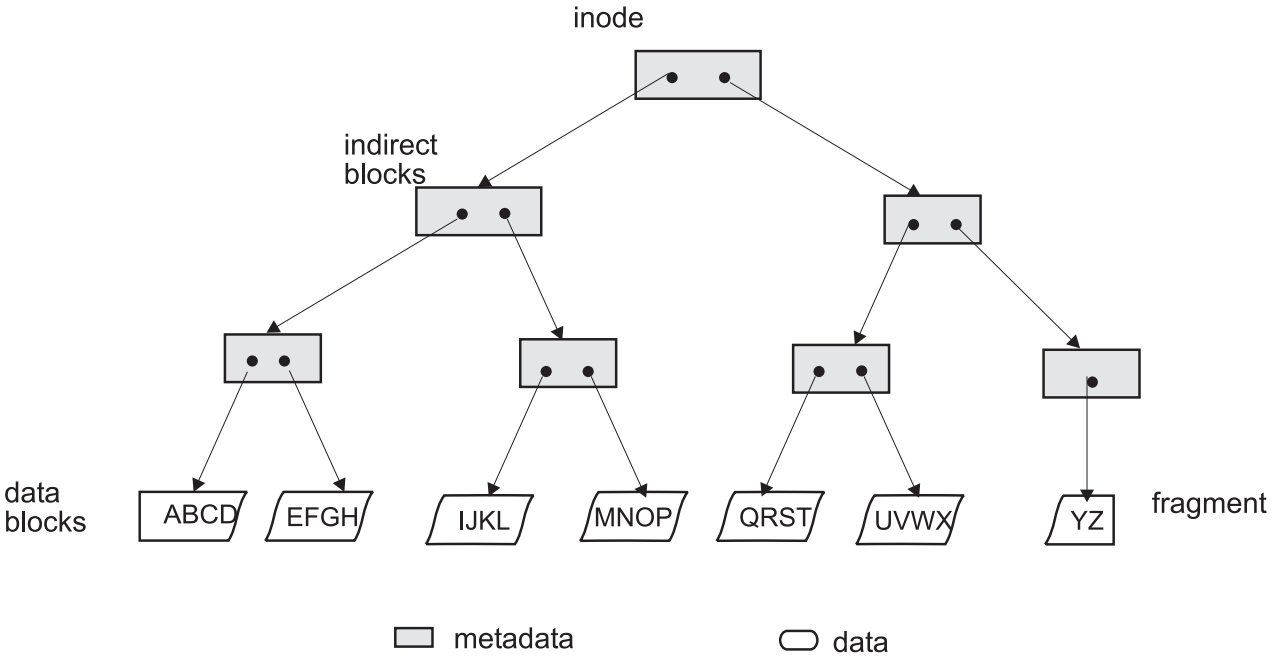


Figure 13. GPFS files have a typical UNIX structure

Note:

1. The maximum number of mounted file systems within a GPFS cluster is 256.
2. See the GPFS FAQ at publib.boulder.ibm.com/infocenter/clresctr/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 of 2,147,484,647.

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

- Root directory of the file system
- *Block allocation map*, 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.

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 may be created if needed. Recovery logs are always replicated and are found through a pointer in the file system descriptor. The file system manager assigns a recovery log to each node accessing the file system.

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

There are certain failure cases where blocks are marked allocated but not part of a file, and this can be recovered by running **mmfsck** online or offline. GPFS always replicates its log. There are two copies of the log for each executing node. Log recovery is run:

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

Note: Any 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

Note: Other nodes may assume token management responsibilities. Refer to *Large system considerations* in *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 are 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 23.

Pinned memory is labeled, *pagepool*. 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

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 may increase I/O performance for file systems if they have these operating characteristics:

- Frequent writes that can be overlapped with application execution
- Reuse of files and sequential reads of a size such that prefetch will benefit the application

Non-pinned memory

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. However, the number of inodes for recently used files is constrained by how much the **maxFilesToCache** parameter exceeds the number of currently open files.

Note: The number of open files can exceed the value defined by the **maxFilesToCache** parameter.

The stat cache contains enough information to respond to inquiries about the file and open it, but not enough information to read from it or write to it. There is sufficient data from the inode to respond to a **stat( )** call (the system call under commands such as **ls -l**). A stat cache entry consumes 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:

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

Note:

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

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 administration 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"
- "GPFS administration commands" on page 86

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

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” on page 88)
- GPFS commands described in the *Commands* section of the *GPFS: Administration and Programming Reference* (see also“GPFS command processing” on page 91)
- 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 initialization can be done automatically as part of the node startup sequence, or manually using the **mmstartup** command to start the daemon. The daemon startup process loads the necessary kernel extensions, if they have not been previously loaded by an earlier instance of the daemon subsequent to the current boot of this node. The initialization sequence 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*. This transition is evident in a message to the GPFS console file (**/var/adm/ras/mmfs.log.latest**). When this state changes from *initializing* to *active*, the daemon is ready to accept mount requests. The message `mmfsd ready` will appear in the GPFS console file.

The mounting of a file system

GPFS file systems are mounted using the operating system's **mount** command, or the GPFS **mmmount** command. GPFS mount processing builds the structures that serve as the 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 will ensure that the file system is ready to be mounted. This includes checking that there are no conflicting utilities being run by the **mmfsck** or **mmcheckquota** commands, for example, and running any necessary log processing to ensure that metadata on the file system is consistent.

On the local node the control structures required for a mounted file system are initialized and the token management function domains are created. In addition, paths to each of the disks 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 is through normal file system calls to the operating system, which are relayed to GPFS if data in a GPFS file system is involved. This uses GPFS code in a kernel extension, which attempts to satisfy the application request using data already in memory. If this can be accomplished, control is returned to the application through the operating system interface. If the request requires resources that are not available at the time, the request is transferred for execution by a daemon thread. The daemon threads wait for work in a system call in the kernel, and are scheduled as necessary. Services available at the daemon level are the acquisition of tokens and disk I/O.

Operating system calls operate on GPFS data during:

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

The open of a GPFS file

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

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

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

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

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

The reading of data

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

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

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

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 we at least have a read token that includes the buffer. After the copying, prefetch is initiated if appropriate.

Tokens available locally but data must be read:

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

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

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

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

Data and tokens must be acquired:

The third, and most complex **read** operation requires that tokens as well as data be acquired on the application node. The kernel code determines that the data is not available locally and sends the message to the daemon waiting after posting the message. The daemon thread determines that it does not have the required tokens to perform the operation. In that case, a token acquire request is sent to the token management server. The requested token specifies a required length of that range of the file, which is needed for this buffer. If the file is being accessed sequentially, a desired range of data, starting at this point of this read and extending to the end of the file, is specified. In the event that no conflicts exist, the desired range will be granted, eliminating the need for token calls on subsequent reads. After the minimum token needed is acquired, the flow proceeds as in step 3 on page 80 (token management).

The writing of 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 technique allows better scheduling of the disk and improved performance. The file system buffers come from the memory allocated based on the **pagepool** parameter in the **mmchconfig** command. Increasing this value may allow more writes to be deferred, which improves performance in certain workloads.

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

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

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

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

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

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

Buffer available in memory:

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

Token available locally but data must be read:

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

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

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

Data and tokens must be acquired:

The third, and most complex path through **write** occurs when neither the buffer nor the token exists at the local node. Prior to the allocation of a buffer, a token is acquired for the area of the file 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” on page 90.

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 but the file system is not mounted, a file system manager is appointed for the task. The **mmchdisk** command and the **mmfsck** command represent two typical types of commands 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

NSD disk access through disk discovery invokes the GPFS shell script **/usr/lpp/mmfs/bin/mmdevdiscover** and if it exists, the user modifiable shell script **/var/mmfs/etc/nsddevices**.

These scripts provide a list of available disk devices that appear in the node's local **/dev** file system. This list is subsequently used by GPFS in determining if a **/dev** device interface on the local node maps to an NSD name recorded in the configuration database.

The process of mapping a **/dev** 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 a 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 **/dev** interface.

If no **/dev** 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 unnecessary to understand the internals of GPFS failure recovery processing. 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 will be 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 will handle all failures prior to completing any recovery.

GPFS recovers from 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 and processing a failure message for the old 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 of the group. After this is complete, the actual recovery of the log of the failed node proceeds. This recovery will rebuild the metadata that was being modified at the time of the failure to a consistent state with the possible exception that blocks may be allocated that are not part of any file and are effectively lost until **mmfsck** is run, online or offline. After log recovery is complete, the locks held by the failed nodes are released for this file system. Completion of this activity for all file systems completes the failure processing. The completion of the protocol allows a failed node to rejoin the cluster.

Cluster configuration data files

GPFS commands save 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 file 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 22. 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 server nodes is inaccessible it can be changed through 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. Some of these files are:

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

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

/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.

Those files are:

.mmbuControl
Contains the control information and the results from the previous invocation of the **mmbackup** command. This includes the node designated as the backup server, nodes designated as backup clients, the number of processes per client node, the completion level (return code), and the total number of inodes processed. This information is used in resuming a partially successful backup.

.mmbuPendingChgs
Contains information regarding a partially successful invocation of the **mmbackup** command, where all changes did not complete. This file is used when the command is reissued with the resume (**-R**) option.

.mmbuPendingDels
Contains information regarding a partially successful invocation of the **mmbackup** command, where all deletions did not complete. This file is used when the command is reissued with the resume (**-R**) option.

.mmbuShadowCheck
Contains a list of all the files that were backed up. This file is used for subsequent incremental backups. The backup restore utility performs a full incremental backup where files deleted from the file system are deleted from the backup copy. In order to determine which files have since been deleted, this file must be available from the previous backup. The size of the shadow file is approximately (1100 * *N*) bytes, where *N* is the number of files in the file system.

The **mmbackup** command may create other files as well. Assume that files whose names begin with **.mmbu** are associated with the **mmbackup** command, and do not manually delete or change them.

Chapter 11. IBM Virtual Shared Disk considerations

On AIX nodes in your cluster, disk subsystem support may be provided through the IBM Virtual Shared Disk component and the IBM Recoverable Virtual Shared Disk component.

- The IBM Virtual Shared Disk component provides disk driver level support for GPFS cluster wide disk accessibility.
- The IBM Recoverable Virtual Shared Disk component provides the capability to fence a node from accessing certain disks, which is a prerequisite for successful recovery of that node. It also provides for transparent failover of disk access in the event of the failure of a disk server.

Software simulation of a SAN is provided by the use of the IBM Virtual Shared Disk component of RSCT. Disks attached in this manner are formatted into virtual shared disks for use by GPFS through the **mmcrvsd** command provided by GPFS or by the **createvsd** command provided by the RSCT subsystem.

The IBM Virtual Shared Disk subsystem supports two methods of external disk access:

- A non-concurrent mode in which only one virtual shared disk server has access to a shared external disk at a given time. A primary and a backup server are defined.
- A concurrent mode in which multiple servers are defined to access the disk concurrently.

The IBM Recoverable Virtual Shared Disk component allows a secondary or backup server to be defined for a logical volume, providing the fencing capabilities required to preserve data integrity in the event of certain system failures. See “Node failure” on page 14. Therefore, the IBM Recoverable Virtual Shared Disk component is required even in the event there are no twin-tailed disks. The *Reliable Scalable Cluster Technology: Managing Shared Disks* manual contains installation, management, and usage information for both the IBM Virtual Shared Disk and the IBM Recoverable Virtual Shared Disk.

Virtual shared disk server considerations

There are several virtual shared disk server considerations that you need to plan for including disk distribution and disk connectivity.

Will your virtual 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 virtual shared disk server, servicing disk requests from other nodes might conflict with the demands of these applications.

The special functions of the GPFS file system manager consume extra processing time. If possible, avoid using a virtual shared disk server as the file system manager. The virtual 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 80.

The actual processing capability required for virtual 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 virtual shared disk server.

Assure that you have sufficient resources to run the IBM Virtual Shared Disk program efficiently. This includes enough buddy buffers of sufficient size to match your file system block size, as well as setting other parameters in the communications subsystem. See the *Reliable Scalable Cluster Technology: Managing Shared Disks* manual for your environment and search on *Performance and tuning considerations for virtual shared disks*.

Disk distribution

Plan how to distribute your disks among the virtual shared disk servers. Two considerations should guide your decision. One involves providing sufficient disks and adapters on the system to yield the required I/O bandwidth. The other involves knowing approximately how much storage capacity you will need for your data. Dedicated virtual shared disk servers should have sufficient disks and adapters to drive the I/O load you expect them to handle. See “Disk I/O” on page 73 for further information on configuring your disk I/O options.

Prepare a list of disks that each virtual shared disk server will be using. This list will be helpful when creating disk descriptors during file system creation. If you have multi-tailed disks, and want to configure for primary and backup virtual shared disk servers (to protect against virtual shared disk server node failure), record the disk device name on the primary server, and the node numbers of the primary and backup servers. For example, if your virtual shared disk servers are nodes 1, 3, 5, and 7:

```
Disk          on Primary node     Backup node
hdisk2               1                3
hdisk3               3                1
hdisk2               5               n/a
hdisk2               7               n/a
```

In this case, nodes 1 and 3 share disks using multi-tailing and will back up each other. However, nodes 5 and 7 will each bear the full responsibility of serving their disks. These are the disks that will be made into virtual shared disks from which your GPFS file system will be constructed.

Disk connectivity

If your disks are capable of twin-tailing and you wish to exploit this capability, you must select an alternate node as the backup virtual shared disk server. See the *Reliable Scalable Cluster Technology: Managing Shared Disks* manual for your environment for help in selecting these nodes.

Virtual shared disk creation considerations

GPFS uses virtual shared disks to access raw logical volumes as if they were local at each of the nodes. Although the *Managing Shared Disks* book is the definitive source for instructions on how to create virtual shared disks, you can have GPFS create them through the **mmcrvsd** command.

For performance reasons, GPFS creates one virtual shared disk for each physical disk specified for the file system, and assigns an optimal partition size based on the disk’s capacity. A virtual shared disk name is also automatically generated. If you want to take advantage of the flexibility available in creating virtual shared disks, follow the instructions in the *Reliable Scalable Cluster Technology: Managing Shared Disks* manual then pass the newly created virtual shared disk to the GPFS file system by specifying the virtual shared disk name (see “Disks for your file system” on page 33).

The **mmcrvsd** 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

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

Alternatively, the name of a virtual shared disk created using AIX and virtual shared disk commands. In this case, the virtual shared disk is registered in the GPFS configuration database for subsequent use by GPFS commands.

PrimaryServer
The name of the primary virtual shared disk server node.

BackupServer
The backup server name.

Disk Usage
What is to be stored on the disk.

dataAndMetadata
Indicates that the disk contains both data and metadata. This is the default.

dataOnly
Indicates that the disk contains data and does not contain metadata.

metadataOnly
Indicates that the disk contains metadata and does not contain data.

descOnly
Indicates that the disk contains no data or metadata and is used solely to keep a copy of the file system descriptor. Such a disk allows file system descriptor quorum to be maintained.

Disk usage considerations:

1. The *DiskUsage* parameter is not utilized by the **mmcrvsd** command but is copied intact to the output file that the command produces. The output file may then be used as input to the **mmcrnsd** command.
2. RAID devices are not well-suited for performing small block writes. Since GPFS metadata writes are often smaller than a full block, you may find using non-RAID devices for GPFS metadata better for performance.

FailureGroup
A number identifying the failure group to which this disk belongs. All disks that are either attached to the same adapter or virtual shared disk server have a common point of failure and should therefore be placed in the same failure group as shown in Figure 14.

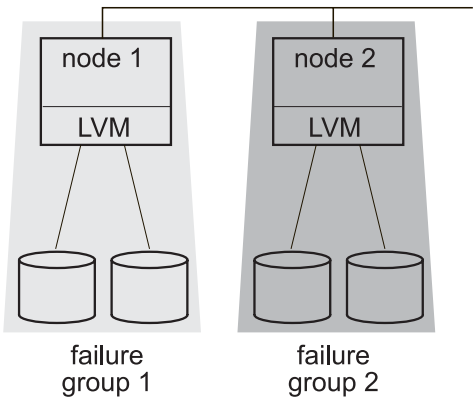


Figure 14. Basic failure groups with servers and disks

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

If you plan to use both twin-tailed disks and replication, assign disks to the failure groups with their primary servers, as shown in Figure 15. This arrangement would assure availability of replicated data if either server failed.

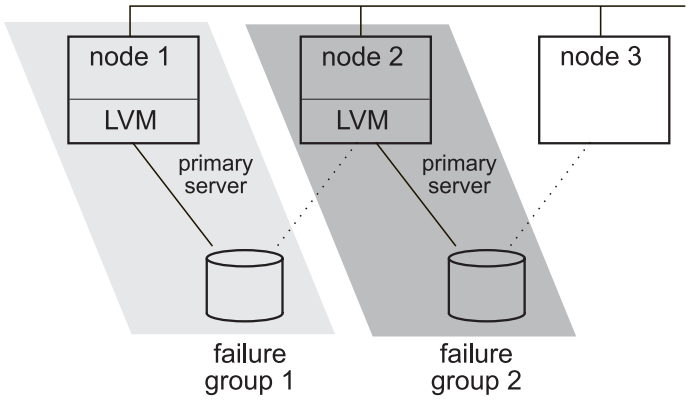


Figure 15. Failure groups with twin-tailed disks

Failure group considerations: The *FailureGroup* parameter is not utilized by the **mmcrvsd** command but is copied intact to the output file that the command produces. The output file may then be used as input to the **mmcrnsd** command.

DesiredName
Specify the name you desire for the virtual shared disk to be created. This name must not already be used by another GPFS disk name, and it must not begin with the reserved string ″gpfs″. If a desired name is not specified, the virtual shared disk is assigned a name according to the convention:

gpfs*NN***vsd**
Where *NN* is a unique non negative integer not used in any prior virtual shared disk. These global disk names must be subsequently used on all GPFS commands. GPFS commands, other than the **mmcrvsd** command, will not accept physical disk device names.

If a desired name is specified on the disk descriptor, **mmcrvsd** uses that name as the basis for the names of the global volume group, local logical volume, and local volume group name according to the convention:

*DesiredName***gvg**
The global volume group

*DesiredName***lv**
The local logical volume

*DesiredName***vg**
The local volume group

If a desired name is not specified on the disk descriptor, **mmcrvsd** assigns the names of the global volume group, local logical volume, and local volume group name according to the convention:

gpfs*NN***gvg**
Where *NN* is a unique non negative integer not used in any prior global volume group named with this convention.

gpfs*NN***lv**
Where *NN* is a unique non negative integer not used in any prior logical volume named with this convention.

gpfs*NN*vg
Where *NN* is a unique non negative integer not used in any prior volume group named with this convention.

StoragePool
Specifies the name of the storage pool that the NSD is assigned to. This field is ignored by the **mmcrnsd** command, and is passed unchanged to the output descriptor file produced by the **mmcrnsd** command.

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

- The physical device or vpath name is replaced with the created virtual shared disk name.
- The primary and backup servers are omitted.
- The *DiskUsage* and *FailureGroup* fields are not changed.

The rewritten disk descriptor file, *DescFile*, can then be used as input to the **mmcrnsd** command. The *DiskUsage* and *FailureGroup* specifications in the disk descriptor are only preserved in the *DescFile* file rewritten by the **mmcrvsd** 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 *DiskUsage* and *FailureGroup* values for a disk can be changed with the **mmchdisk** command. The virtual shared disk name cannot be changed.

Virtual shared disk server and disk failure

One means of data protection is the use of a RAID controller, which masks disk failures with parity disks. An ideal configuration is shown in Figure 16, where a RAID device is twin-tailed to two nodes. This protects against server failure as well.

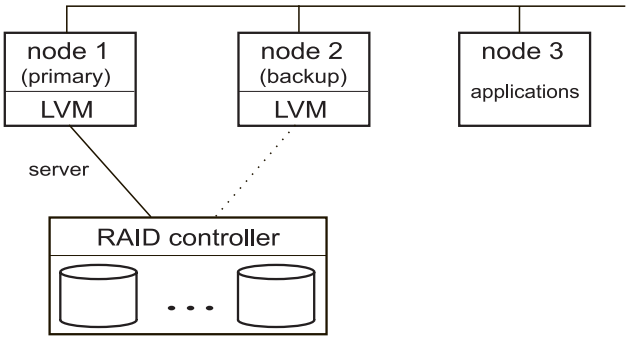


Figure 16. Primary node serving RAID device

If node 1, the primary server, fails, its responsibilities are assumed by node 2, the backup server, as shown in Figure 17 on page 100.

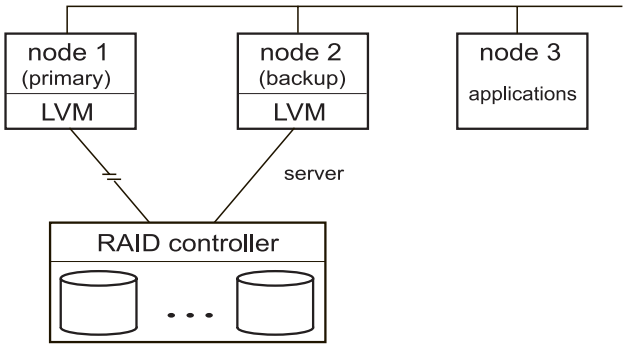


Figure 17. Backup node serving RAID device

If your disks are SAN-attached to the virtual shared disk servers, an ideal configuration is shown in Figure 18.

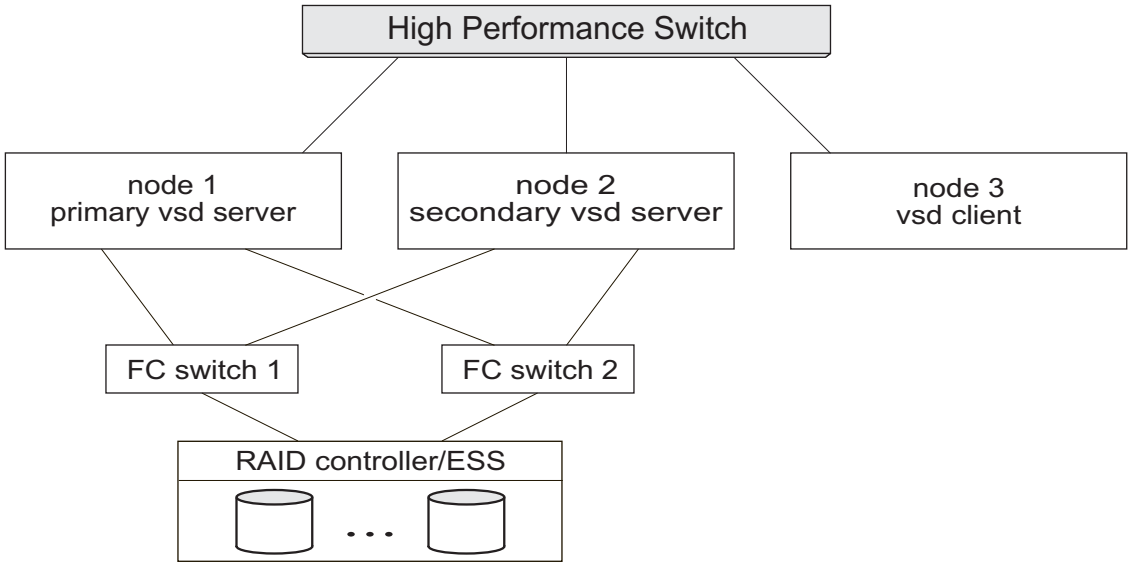


Figure 18. RAID/ESS Controller multi-tailed to the primary and secondary virtual shared disk servers

Another means of data protection is through the use of concurrent virtual shared disks, as shown in Figure 19 on page 101. Concurrent disk access allows you to use multiple servers to satisfy disk requests by taking advantage of the concurrent disk access environment supplied by AIX. For further information regarding concurrent virtual shared disks, see the *Reliable Scalable Cluster Technology: Managing Shared Disks* manual for your environment.

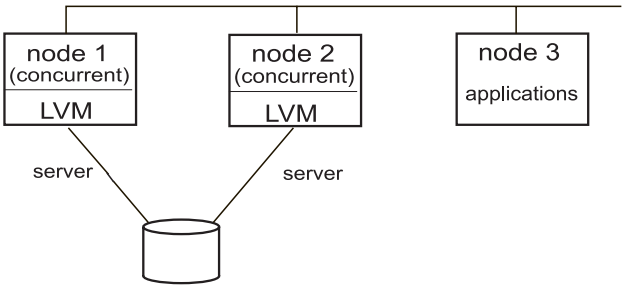


Figure 19. Concurrent node serving device

You can also protect your file system against disk failure by *mirroring data* at the logical volume manager (LVM) level, writing the data twice to two different disks. The addition of twin-tailed disks to such a configuration adds protection against server failure by allowing the IBM Recoverable Virtual Shared Disk program to route requests through a backup server.

Chapter 12. 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 if a file system is controlled by GPFS has been provided.

For more information, see the following topics:

- "Exceptions to Open Group technical standards"
- "Determining if a file system is controlled by GPFS"
- "GPFS exceptions and limitations to NFS V4 ACLs" on page 104

Exceptions to Open Group technical standards

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

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

1. **exact mtime**
2. **mtime**
3. **ctime**
4. **atime**

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

These values will be accurate on a node right after it accesses or modifies a file, but may not be accurate for a short while when a file is accessed or modified on some other node.

If 'exact mtime' is specified for a file system (using the **mmcrfs** or **mmchfs** commands with the **-E yes** flag), the **mtime** and **ctime** values are always correct by the time the **stat()** call gives its answer. If 'exact mtime' is not specified, these values will be accurate after a couple of minutes, to allow the synchronization daemons to propagate the values to all nodes. Regardless of whether 'exact mtime' is specified, the **atime** value will be accurate after a couple of minutes, to allow for all the synchronization daemons to propagate changes.

Alternatively, you may use the GPFS calls, **gpfs_stat()** and **gpfs_fstat()** to return exact **mtime** and **atime** values.

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

Determining if a file system is controlled by GPFS

A file system is controlled by GPFS if the **f_type** field in the **statfs** structure returned from a **statfs()** or **fstatfs()** call has the value 0x47504653, which is the ASCII characters 'GPFS'.

This constant is in the **gpfs.h** file, with the name **GPFS_SUPER_MAGIC**. If an application includes **gpfs.h**, it can compare **f_type** to **GPFS_SUPER_MAGIC** to determine if the file system is controlled by GPFS.

GPFS exceptions and limitations to NFS V4 ACLs

GPFS has exceptions and limitations to the NFS V4 ACLs, which are listed in this topic.

The exceptions and limitations include:

1. Alarm type ACL entries are not supported.
2. Audit type ACL entries are not supported.
3. Inherit entries (**FileInherit** and **DirInherit**) are always propagated to all child subdirectories. The NFS V4 **ACE4_NO_PROPAGATE_INHERIT_ACE** flag is not supported.
4. Although the NFS V4 ACL specification provides separate controls for **WRITE** and **APPEND**, GPFS will not differentiate between the two. Either both must be specified, or neither can be.
5. Similar to **WRITE** and **APPEND**, NFS V4 allows for separate **ADD_FILE** and **ADD_SUBDIRECTORY** controls. In most cases, GPFS will allow these controls to be specified independently. In the special case where the file system object is a directory and one of its ACL entries specifies both **FileInherit** and **DirInherit** flags, GPFS cannot support setting **ADD_FILE** without **ADD_SUBDIRECTORY** (or the other way around). When this is intended, we suggest creating separate **FileInherit** and **DirInherit** entries.
6. Some types of access for which NFS V4 defines controls do not currently exist in GPFS. For these, ACL entries will be accepted and saved, but since there is no corresponding operation they will have no effect. These include **READ_NAMED**, **WRITE_NAMED**, and **SYNCHRONIZE**.
7. AIX requires that **READ_ACL** and **WRITE_ACL** always be granted to the object owner. Although this contradicts *NFS Version 4 Protocol*, it is viewed that this is an area where users would otherwise erroneously leave an ACL that only privileged users could change. Since ACLs are themselves file attributes, **READ_ATTR** and **WRITE_ATTR** are similarly granted to the owner. Since it would not make sense to then prevent the owner from accessing the ACL from a non-AIX node, GPFS has implemented this exception everywhere.
8. AIX does not support the use of special name values other than **owner@**, **group@**, and **everyone@**. Therefore, these are the only valid special name for use in GPFS NFS V4 ACLs as well.
9. NFS V4 allows ACL entries that grant users (or groups) permission to change the owner or owning group of the file (for example, with the **chown** command). For security reasons, GPFS now restricts this so that non-privileged users may only **chown** such a file to themselves (becoming the owner) or to a group that they are a member of.
10. GPFS does not support NFS V4 exporting GPFS file systems from Linux nodes. NFS V3 is acceptable.

For more information about GPFS ACLs and NFS export, see *Managing GPFS access control lists and NFS export* in *General Parallel File System: Administration and Programming Reference*.

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 http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.addinfo.doc/access.html.

Keyboard navigation

This product uses standard Microsoft Windows navigation keys.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility:

`http://www.ibm.com/able`

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 (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

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

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Intellectual Property Law
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 Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at ″Copyright and trademark information″ at www.ibm.com/legal/copytrade.shtml

Intel®, Intel Inside® (logos), MMX and Pentium® are trademarks of Intel Corporation in the United States, other countries, or both.

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, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat, the Red Hat “Shadow Man” logo, and all Red Hat-based trademarks and logos are trademarks or registered trademarks of Red Hat, Inc., in the United States and other countries.

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 process of transferring all control of the ESS to a single cluster in the ESS when the other cluster in the ESS fails. See also *cluster*. (2) The routing of all transactions to a second controller when the first controller fails. See also *cluster*. (3) The assumption of file system duties by another node when a node fails.

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.

IBM Virtual Shared Disk. The subsystem that allows application programs running on different nodes to access a logical volume as if it were local to each node. The logical volume is local to only one of the nodes (the server node).

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.

L

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

Logical Volume Manager (LVM). A set of system commands, library routines, and other tools that allow the user to establish and control logical volume (LVOL) storage. The LVM maps data between the logical view of storage space and the physical disk drive module (DDM).

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.

Q

quorum node. A node in the cluster that is counted to determine whether a quorum exists.

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

quota management. The allocation of disk blocks to the other nodes writing to the file system, and comparison of the allocated space to quota limits at regular intervals.

R

Redundant Array of Independent Disks (RAID). A collection of two or more disk physical drives that present to the host an image of one or more logical disk drives. In the event of a single physical device failure, the data can be read or regenerated from the other disk drives in the array due to data redundancy.

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

replication. The process of maintaining a defined set of data in more than one location. Replication involves copying designated changes for one location (a source) to another (a target), and synchronizing the data in both locations.

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.

Serial Storage Architecture (SSA). An American National Standards Institute (ANSI) standard, implemented by IBM, for a high-speed serial interface that provides point-to-point connection for peripherals, such as storage arrays.

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.

SSA. See Serial Storage Architecture.

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 shared disk. See *IBM Virtual Shared Disk*.

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

Readers' comments – We'd like to hear from you

General Parallel File System
Concepts, Planning, and Installation Guide
Version 3 Release 2.1

Publication No. GA76-0413-02

We appreciate your comments about this publication. Please comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. The comments you send should pertain to only the information in this manual or product and the way in which the information is presented.

For technical questions and information about products and prices, please contact your IBM branch office, your IBM business partner, or your authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you. IBM or any other organizations will only use the personal information that you supply to contact you about the issues that you state on this form.

Comments:

Thank you for your support.

Submit your comments using one of these channels:

- Send your comments to the address on the reverse side of this form.
- Send your comments via e-mail to: mhvrcfs@us.ibm.com

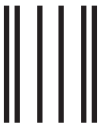
If you would like a response from IBM, please fill in the following information:

Name	Address
Company or Organization	
Phone No.	E-mail address

Readers' Comments — We'd Like to Hear from You
GA76-0413-02

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 58HA, Mail Station P181
2455 South Road
Poughkeepsie, NY
12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5724-N94
5765-G66

GA76-0413-02