



An Introduction to GPFS Version 3.2

September, 2007

Scott Fadden

IBM Corporation

Contents	
Overview	2
What is GPFS?	3
The file system	3
Application interfaces	4
Performance and scalability	4
Administration	5
Data availability	6
Information lifecycle management (ILM)	7
Cluster configurations	8
Shared disk	9
Network-based block IO	9
Sharing data between clusters	12
What's new in GPFS Version 3.2	14
Summary	16

Overview

This paper provides an overview of IBM General Parallel File System™ (GPFS™) Version 3, Release 2 for AIX® and Linux®. It includes concepts key to understanding, at a high level, available features and functionality.

This paper covers core GPFS concepts including the high-performance file system, direct storage area network (SAN) access, network based block I/O information lifecycle management (ILM) tools and new features including multiple NSD servers, clustered NFS and more scalable ILM tools supporting migration to other media types.

The goal of this paper is to provide an introduction to GPFS features and terminology. For a more detailed description of any of these topics you should reference the product documentation See: [GPFS 3.2 documentation](#).

This paper is based on the latest release of GPFS though much of the information applies to prior releases as well. It is assumed that the reader has a basic knowledge of clustering and storage networks.

What is GPFS?

The IBM General Parallel File System (GPFS) provides unmatched performance and reliability with scalable access to critical file data. GPFS distinguishes itself from other cluster file systems by providing concurrent high-speed file access to applications executing on multiple nodes of an AIX cluster, a Linux cluster, or a heterogeneous cluster of AIX and Linux nodes. In addition to providing file storage capabilities, GPFS provides storage management, information life cycle tools, centralized administration and allows for shared access to file systems from remote GPFS clusters.

GPFS provides scalable high-performance data access from a two node cluster providing a high availability platform supporting a database application, for example, to 2,000 nodes or more used for applications like modeling weather patterns. Up to 512 Linux nodes or 128 AIX nodes with access to one or more file systems are supported as a general statement and larger configurations exist by special arrangements with IBM. The largest existing configurations exceed 2,000 nodes. GPFS has been available on AIX since 1998 and Linux since 2001. It has proven time and again on some of the world's most powerful supercomputers¹ to provide efficient use of disk bandwidth.

GPFS was designed from the beginning to support high performance computing (HPC) and has been proven very effective for a variety of applications. It is installed in clusters supporting relational databases, digital media and scalable file serving. These applications are used across many industries including financial, retail and government applications. Being tested in very demanding large environments makes GPFS a solid solution for any size application.

GPFS supports various system types including the IBM System p™ family and machines based on Intel® or AMD processors such as an IBM System x™ environment. Supported operating systems for GPFS Version 3.2 include AIX V5.3 and selected versions of Red Hat and SUSE Linux distributions.

This paper introduces a number of GPFS features and describes core concepts. This includes the file system, high availability features, information lifecycle management (ILM) tools and various cluster architectures.

The file system

A GPFS file system is built from a collection of disks which contain the file system data and metadata. A file system can be built from a single disk or

¹ Four clusters in the top 10 as of July 7, 2006 - Source: Top 500 Super Computer Sites: <http://www.top500.org/>

contain thousands of disks, storing Petabytes of data. A GPFS cluster can contain up to 256 mounted file systems. There is no limit placed upon the number of simultaneously opened files within a single file system. As an example, current GPFS customers are using single file systems up to 2PB in size and others containing tens of millions of files

Application interfaces

Applications can access files through standard UNIX® file system interfaces or through enhanced interfaces available for parallel programs. Parallel and distributed applications can be scheduled on GPFS clusters to take advantage of the shared access architecture. This makes GPFS a key component in many grid-based solutions. Parallel applications can concurrently read or update a common file from multiple nodes in the cluster. GPFS maintains the coherency and consistency of the file system using a sophisticated byte level locking, token (lock) management and logging.

In addition to standard interfaces GPFS provides a unique set of extended interfaces which can be used to provide high performance for applications with demanding data access patterns. These extended interfaces are more efficient for traversing a file system, for example, and provide more features than the standard POSIX interfaces.

Performance and scalability

GPFS provides unparalleled performance especially for larger data objects and excellent performance for large aggregates of smaller objects. GPFS achieves high performance I/O by:

- Striping data across multiple disks attached to multiple nodes.
- Efficient client side caching.
- Supporting a large block size, configurable by the administrator, to fit I/O requirements.
- Utilizing advanced algorithms that improve read-ahead and write-behind file functions.
- Using block level locking based on a very sophisticated scalable token management system to provide data consistency while allowing multiple application nodes concurrent access to the files.

GPFS recognizes typical access patterns like sequential, reverse sequential and random and optimizes I/O access for these patterns.

GPFS token (lock) management coordinates access to shared disks ensuring the consistency of file system data and metadata when different

nodes access the same file. GPFS has the ability for multiple nodes to act as token managers for a single file system. This allows greater scalability for high transaction workloads.

Along with distributed token management, GPFS provides scalable metadata management by allowing all nodes of the cluster accessing the file system to perform file metadata operations. This key and unique feature distinguishes GPFS from other cluster file systems which typically have a centralized metadata server handling fixed regions of the file namespace. A centralized metadata server can often become a performance bottleneck for metadata intensive operations and can represent a single point of failure. GPFS solves this problem by managing metadata at the node which is using the file or in the case of parallel access to the file, at a dynamically selected node which is using the file.

Administration

GPFS provides an administration model that is consistent with standard AIX and Linux file system administration while providing extensions for the clustering aspects of GPFS. These functions support cluster management and other standard file system administration functions such as quotas, snapshots, and extended access control lists.

GPFS provides functions that simplify cluster-wide tasks. A single GPFS command can perform a file system function across the entire cluster and most can be issued from any node in the cluster. These commands are typically extensions to the usual AIX and Linux file system commands.

Rolling upgrades allow you to upgrade individual nodes in the cluster while the file system remains online. With GPFS Version 3.1 you could mix GPFS 3.1 nodes with different patch levels. Continuing that trend in GPFS Version 3.2 you can run a cluster with a mix of GPFS Version 3.1 and GPFS Version 3.2 nodes.

Quotas enable the administrator to control and monitor file system usage by users and groups across the cluster. GPFS provides commands to generate quota reports including user, group and fileset inode and data block usage. In addition to traditional quota management, GPFS has an API that provides high performance metadata access enabling custom reporting options on very large numbers of files.

A snapshot of an entire GPFS file system may be created to preserve the file system's contents at a single point in time. This is a very efficient mechanism because a snapshot contains a map of the file system at the time it was taken and a copy of only the file system data that has been changed since the snapshot was created. This is done using a copy-on-write technique. The snapshot function allows a backup program, for example, to run concurrently with user updates and still obtain a consistent copy of the file

system as of the time that the snapshot was created. Snapshots provide an online backup capability that allows files to be recovered easily from common problems such as accidental deletion of a file.

An SNMP interface is introduced in GPFS Version 3.2 to allow monitoring by network management applications. The SNMP agent provides information on the GPFS cluster and generates traps in the event a file system is mounted, modified or if a node fails. In GPFS Version 3.2 the SNMP agent runs only on Linux. You can monitor a mixed cluster of AIX and Linux nodes as long as the agent runs on a Linux node.

GPFS provides support for the Data Management API (DMAPI) interface which is IBM's implementation of the X/Open data storage management API. This DMAPI interface allows vendors of storage management applications such as IBM Tivoli® Storage Manager (TSM) to provide Hierarchical Storage Management (HSM) support for GPFS.

GPFS enhanced access control protects directories and files by providing a means of specifying who should be granted access. On AIX, GPFS supports NFS V4 access control lists (ACLs) in addition to traditional ACL support. Traditional GPFS ACLs are based on the POSIX model. Access control lists (ACLs) extend the base permissions, or standard file access modes, of read (r), write (w), and execute (x) beyond the three categories of file owner, file group, and other users, to allow the definition of additional users and user groups. In addition, GPFS introduces a fourth access mode, control (c), which can be used to govern who can manage the ACL itself.

In addition to providing application file service GPFS file systems may be exported to clients outside the cluster through NFS or Samba. GPFS has been used for a long time as the base for a scalable NFS file service infrastructure. Now that feature is integrated in GPFS Version 3.2 and is called clustered NFS. Clustered NFS provides all the tools necessary to run a GPFS Linux cluster as a scalable NFS file server. This allows a GPFS cluster to provide scalable file service by providing simultaneous access to a common set of data from multiple nodes. The clustered NFS tools include monitoring of file services, load balancing and IP address fail over.

Data availability

GPFS is fault tolerant and can be configured for continued access to data even if cluster nodes or storage systems fail. This is accomplished through robust clustering features and support for data replication.

GPFS continuously monitors the health of the file system components. When failures are detected appropriate recovery action is taken automatically. Extensive logging and recovery capabilities are provided which maintain metadata consistency when application nodes holding locks or performing services fail. Data replication is available for journal logs, metadata and data.

Replication allows for continuous operation even if a path to a disk or a disk itself fails.

GPFS Version 3.2 further enhances clustering robustness with connection retries. If the LAN connection to a node fails GPFS will automatically try and reestablish the connection before making the node unavailable. This provides for better uptime in environments experiencing network issues.

Using these features along with a high availability infrastructure ensures a reliable enterprise storage solution.

Information lifecycle management (ILM)

GPFS is designed to help you to achieve data lifecycle management efficiencies through policy-driven automation and tiered storage management. The use of storage pools, filesets and user-defined policies provide the ability to better match the cost of your storage resources to the value of your data.

Storage pools allow you to create groups of disks within a file system. Using storage pools you can create tiers of storage by grouping disks based on performance, locality or reliability characteristics. For example, one pool could be high performance fibre channel disks and another more economical SATA storage. These types of storage pools are called internal storage pools where all of the data management is done within GPFS. A new feature in GPFS Version 3.2 is external storage pools. An external storage pool is an interface that GPFS uses to interact with an external storage management application. When moving data to an external pool GPFS handles all the metadata processing then hands the data to the external application for storage on alternate media, a tape library for example. Data can be retrieved from the external storage pool on demand, as a result of an application opening a file or data can be retrieved in a batch operation.

A fileset is a sub-tree of the file system namespace and provides a way to partition the namespace into smaller, more manageable units. Filesets provide an administrative boundary that can be used to set quotas and be specified in a user defined policy to control initial data placement or data migration. Data in a single fileset can reside in one or more storage pools. Where the file data resides and how it is migrated is based on a set of rules in a user defined policy.

There are two types of user defined policies in GPFS: File placement and File management. File placement policies determine which storage pool file data is initially placed in. File placement rules are determined by attributes known when a file is created such as file name, user, group or the fileset. An example may include "place all files *.avi onto the platinum storage pool", 'place all files created by the CEO on the gold storage pool', or 'place all files in the fileset 'development' in the bronze pool'.

Once files exist in a file system, file management policies allow you to move, mirror or delete files. You can use file management policies to move data from one pool to another without changing the file's location in the directory structure. They can be used to change the replication (mirroring) status at the file level, allowing fine grained control over the space used for data availability. You can use a policy that says: 'replicate all files in /database/payroll which have the extension *.dat and are greater than 1 MB in size to storage pool #2'. In addition, file management policies allow you to prune the file system, deleting files as defined by policy rules. File management policies can use more attributes of a file than placement policies because once a file exists there is more known about the file. In addition to the file placement attributes you can now utilize attributes such as last access time, size of the file or a mix of user and file size. This may result in policies like: 'delete all files named *.temp not accessed in 180 days', 'move all files that are larger than 2 GB to pool2', or 'migrate all files owned by Sally that are larger than 4GB to the SATA storage pool'. Rules can include attributes related to a pool instead of a single file using the *threshold* option. The threshold option is new in GPFS version 3.2. Using thresholds you can create a rule that moves files out of the high performance pool if it is more than 80% full, for example. The threshold option comes with the ability to set high low and pre-migrate thresholds. This means that GPFS begins migrating data at the high threshold, until the low threshold is reached. If a pre-migrate threshold is set GPFS begins copying data until the pre-migrate threshold is reached. This allows the data to continue to be accessed in the original pool until it is quickly deleted to free up space the next time the high threshold is reached.

Policy rule syntax is based on SQL and supports multiple complex statements in a single rule enabling powerful policies. Multiple levels of rules can be applied because the complete policy rule set is evaluated for each file when the policy engine executes.

The most important step in file management operations is processing the file metadata. The GPFS high performance metadata scan interface allows you to efficiently process the metadata for billions of files. Once the candidate list of files is identified, data movement operations can be done by multiple nodes in the cluster. New in GPFS 3.2 is the ability to spread rule evaluation and data movement responsibilities over multiple nodes in the cluster. This is the next step in improving the scalability of the GPFS ILM tool set.

Cluster configurations

GPFS supports a variety of cluster configurations independent of which file system features you require. Cluster configuration options can be characterized into three categories:

- Shared disk
- Network block I/O

- Sharing data between clusters.

Shared disk

A shared disk cluster is the most basic environment. In this configuration, the storage is SAN attached to all machines in the cluster as shown in Figure 1.

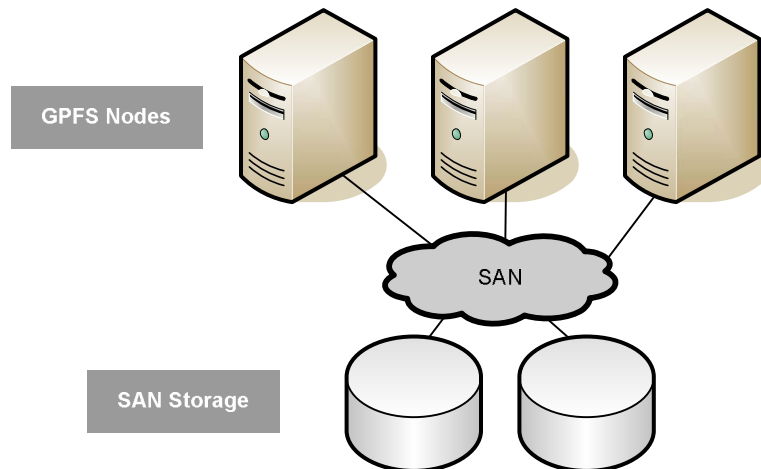


Figure 1: SAN Attached Storage

Figure 1 illustrates a fibre channel SAN. The nodes are connected to the storage using the SAN and to each other using a LAN. Data used by applications flows over the SAN and control information flows among the GPFS instances in the cluster over the LAN.

This configuration is optimal when all nodes in the cluster need the highest performance access to the data. For example, this is a good configuration for providing network file service to client systems using clustered NFS or Samba, high-speed data access for digital media applications or a grid infrastructure for analytics.

Network-based block IO

In some environments, where every node in the cluster cannot be attached to the SAN, GPFS makes use of an IBM provided network block device capability. GPFS provides a block level interface over the LAN called Network Shared Disk (NSD). Whether using NSD or a direct attachment to the SAN the mounted file system looks the same to the application, GPFS transparently handles I/O requests.

GPFS clusters use NSD to provide high speed data access to applications running on LAN attached nodes. Data is served to these client nodes from an NSD server. In this configuration, disks are SAN attached only to the NSD servers. Each NSD server is attached to all or a portion of the disk collection. Starting with GPFS Version 3.2 you can define up to eight NSD servers per disk and it is recommended that at least two NSD servers serve each disk to avoid a single point of failure.

GPFS uses a communications interface for the transfer of control information and data to NSD clients. These communication interfaces need not be dedicated to GPFS; but should provide sufficient bandwidth to meet your GPFS performance expectations and for applications which share the bandwidth. GPFS has the ability to designate separate IP interfaces for intra-cluster communication and the public network. This provides for a clearly defined separation of communication traffic and allows you to increase the throughput and number of nodes in a GPFS cluster. By allowing access to the same disk from multiple subnets the NSD clients no longer have to be on a single physical network. For example you can place separate clients onto separate networks with subsets of the NSD servers accessing a common set of disks so not all NSD servers need to serve all clients. This can reduce networking hardware and support costs and provide greater scalability.

To enable high speed NSD communication GPFS supports 1Gbit and 10 Gbit Ethernet, IBM eServer™ pSeries® High Performance Switch (HPS), InfiniBand and Myrinet for control and data communications. New in GPFS Version 3.2 on Linux a native InfiniBand protocol built on the RDMA verbs API is supported for NSD communication. The native protocol provides much higher performance than TCP/IP over InfiniBand making better use of existing infrastructure.

An example of the NSD server model is shown in Figure 2.

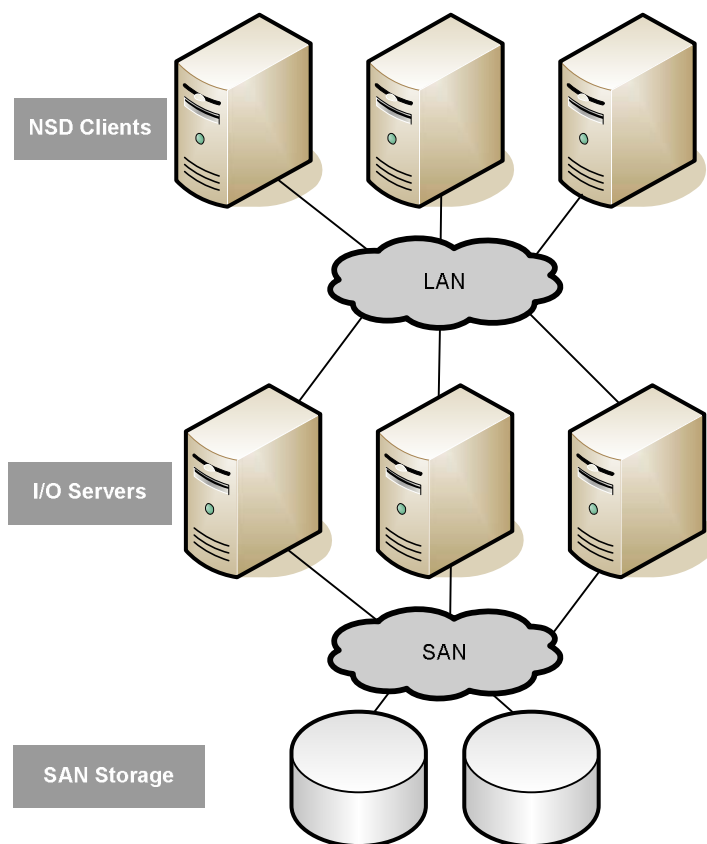


Figure 2: Network block IO

In this configuration, a subset of the total node population is defined as NSD server nodes. The NSD Server is responsible for the abstraction of disk data blocks across an IP-based network. The fact that the disks are remote is transparent to the application. Figure 2 shows an example of a configuration where a set of compute nodes are connected to a set of NSD servers using a high-speed interconnect or an IP-based network such as Ethernet. In this example, data to the NSD servers flows over the SAN and both data and control information to the clients flow across the LAN.

The choice of how many nodes to configure as NSD servers is based on individual performance requirements and the capabilities of the storage subsystem. High bandwidth LAN connections should be used for clusters requiring significant data transfer. This can include 1Gbit, 10 Gbit, the use of link aggregation (Etherchannel or bonding) or higher performance networks such as the HPS or InfiniBand.

The choice between SAN attachment and network block I/O is a performance and economic one. In general, using a SAN provides the highest

performance; but the cost and management complexity of SANs for large clusters is often prohibitive. In these cases network block I/O provides an option.

Network block I/O is well suited to grid computing and clusters with sufficient network bandwidth between the NSD servers and the clients. For example, an NSD based grid is effective for Web applications, supply chain management or modeling weather patterns.

Sharing data between clusters

GPFS allows you to share data across clusters. You can allow other clusters to access one or more of your file systems and you can mount file systems that belong to other GPFS clusters for which you have been authorized. A multi-cluster environment allows the administrator to permit access to specific file systems from another GPFS cluster. This feature is intended to allow clusters to share data at higher performance levels than file sharing technologies like NFS or Samba. It is not intended to replace such file sharing technologies which are tuned for desktop access or for access across unreliable network links. A multi-cluster environment requires a trusted kernel at both the owning and sharing clusters.

Multi-cluster capability is useful for sharing across multiple clusters within a physical location or across locations. Clusters are most often attached using a LAN, but in addition the cluster connection could include a SAN. Figure 3 illustrates a multi-cluster configuration with both LAN and mixed LAN and SAN connections.

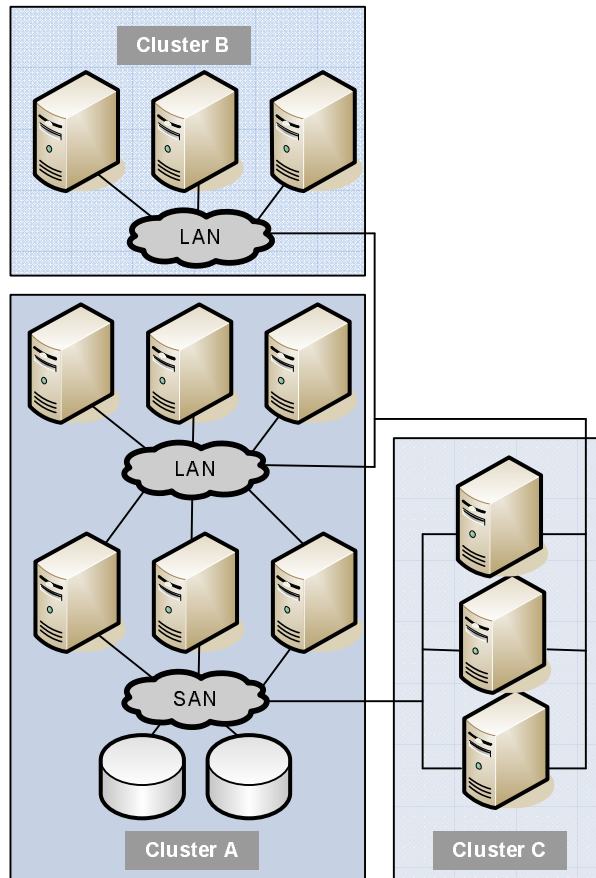


Figure 3: Multi-cluster

In Figure 3, Cluster B and Cluster C need to access the data from Cluster A. Cluster A owns the storage and manages the file system. It may grant access to file systems which it manages to remote clusters such as Cluster B and Cluster C. In this example, Cluster B and Cluster C do not have any storage but that is not always true. They could own file systems which may or may not be accessible outside their cluster. Commonly in the case where a cluster does not own storage, the nodes are grouped into clusters for ease of management. When the remote clusters need access to the data, they mount the file system by contacting the owning cluster and passing required security checks. Cluster B accesses the data through an extension of the NSD network utilizing NSD protocols. Cluster C accesses data through an extension of the storage network and controls flow through an IP network shown in Figure 3. Both types of configurations are possible.

Multi-cluster environments are well suited to sharing data across different groups for collaborative computing between departments or separate locations for example.

What's new in GPFS Version 3.2

For those who are familiar with GPFS 3.1 this section provides a list of what is new in GPFS Version 3.2. This list is not intended to provide details of the new features, for more details refer to the [GPFS documentation](#).

- Performance enhancements include support: for parallel file system defragmentation and large pagepool support:
 - Defragmentation of a file system can now be run in parallel on all nodes in a cluster.
 - Maximum pagepool size is increased to 256 GB.
 - Fine grained directory locking improves multi-node file creation performance into a single directory (to be introduced in an APAR following the GPFS version 3.2 general release).
- GPFS Version 3.2 on Linux includes a native InfiniBand protocol built on the RDMA verbs API.. GPFS uses RDMA to transfer data directly between the NSD client memory and the NSD server memory instead of sending and receiving the data over a TCP socket. Using RDMA can improve bandwidth and decrease CPU utilization.
- SNMP is supported for monitoring a GPFS cluster and generating notification traps. Currently the SNMP process must be run on a Linux node.
- GPFS Version 3.2 is the first major version that supports rolling upgrades of consecutive releases. Rolling upgrades enable you to install new GPFS code one node at a time without shutting down GPFS on other nodes. Some new features become available on each node as soon as the node is upgraded, while other features become available once all participating nodes have been upgraded. Backward compatibility is available to support multi-cluster environments. This allows the administrator to upgrade the local cluster while still allowing mounts from remote nodes in other clusters that have not been upgraded yet.
- You can now designate up to eight NSD servers to simultaneously serve I/O requests to a single disk. Each of these NSD servers must have physical access to the same LUN. This allows different servers to serve I/O to different non-intersecting sets of clients for a variety of reasons, such as load balancing on the server, network partitioning by balancing the load on different networks, or workload partitioning. Multiple NSD server functions require all (peer) NSD servers to be part of the same GPFS cluster. The existing subnet functions in GPFS determine which NSD server should serve a particular client.
- When a socket connection breaks due to a network failure, GPFS attempts to re-establish the connection rather than immediately initiating node expulsion procedures.

- GPFS 3.2 extends ILM functionality to integrate with external storage management applications. With GPFS Version 3.2 a single set of policies can be used to move data across different storage pools within a file system and move data from GPFS to an external storage management application for storage on tape, for example. Additional enhancements include the ability to use policies while restoring data, pool thresholds to initiate file data movement and scalable rule evaluation across multiple nodes. GPFS 3.2 enables the policy code to run in parallel across all nodes in the home cluster that have the file system mounted. The policy evaluation can then scale with the size of the cluster.
- GPFS Version 3.2 offers Clustered NFS which is a set of features and tools that support a high availability solution for NFS exporting file systems. Using Clustered NFS, some or all of the nodes in the GPFS cluster can export the same file systems to the NFS clients. This support is for systems with Linux only. The clustered NFS feature includes:
 - Monitoring: Every node in the NFS cluster runs an NFS monitoring utility that monitors GPFS, the NFS server and networking components on the node. After failure detection, based on customer configuration, the monitoring utility may invoke a failover.
 - Failover: The automatic failover procedure transfers the NFS serving load from the failing node to another node in the NFS cluster. The failure is managed by the GPFS cluster, including NFS server IP address failover and file system lock and state recovery.
 - Load balancing: Load balancing is IP address based. The IP address is the load unit that can be moved from one node to another for failure or load balancing needs. This solution supports a failover of all the node's load as one unit to another node. However, if no locks are outstanding, individual IP addresses can be moved to other nodes for load balancing purposes.
- GPFS can use the SCSI-3 (Persistent Reserve) standard to provide fast failover with improved recovery times. To exploit this functionality the file system has to be created on SCSI-3 disks. This functionality is available only on AIX.

Summary

With unparalleled scalability and performance, GPFS is the file management solution for demanding I/O environments such as digital media with support for high bandwidth streaming data. It is a cornerstone of grid applications supporting market research, financial analytics, data mining and other large distributed workloads. With clustered NFS GPFS now includes scalable NFS file services for enterprise-wide user file storage and can also support FTP and Samba servers. The proven GPFS high-availability features provide a solid infrastructure for mission-critical relational database applications, clustered Web or application services or any application that requires scalable, highly available storage infrastructure.

You can get details on any of these features in the [GPFS v3.2 documentation](#) available at:

<http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfsbooks.html>

See the [GPFS FAQ](#) for a current list of tested machines and Linux distribution levels and supported interconnects at:

http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.html

For more information on IBM General Parallel File System, visit ibm.com/servers/eserver/clusters/software/gpfs.html or contact your IBM representative.



© IBM Corporation 2007

IBM Corporation
Marketing Communications
Systems Group
Route 100
Somers, New York 10589

Produced in the United States of America
August 2007
All Rights Reserved

This document was developed for products and/or services offered in the United States. IBM may not offer the products, features, or services discussed in this document in other countries.

The information may be subject to change without notice. Consult your local IBM business contact for information on the products, features and services available in your area.

All statements regarding IBM's future directions and intent are subject to change or withdrawal without notice and represent goals and objectives only.

IBM, the IBM logo, AIX, eServer, General Purpose File System, GPFS, pSeries, System p, System x, Tivoli are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both. A full list of U.S. trademarks owned by IBM may be found at <http://www.ibm.com/legal/copytrade.shtml>.

UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Linux is a trademark of Linus Torvalds in the United States, other countries or both.

Intel is a registered trademark of Intel Corporation in the United States and/or other countries.

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

Information concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of the non-IBM products should be addressed with the suppliers.

When referring to storage capacity, 1 TB equals total GB divided by 1024; accessible capacity may be less.

The IBM home page on the Internet can be found at <http://www.ibm.com>.

The IBM System p home page on the Internet can be found at <http://www.ibm.com/systems/p>.

CLW03005-USEN-00