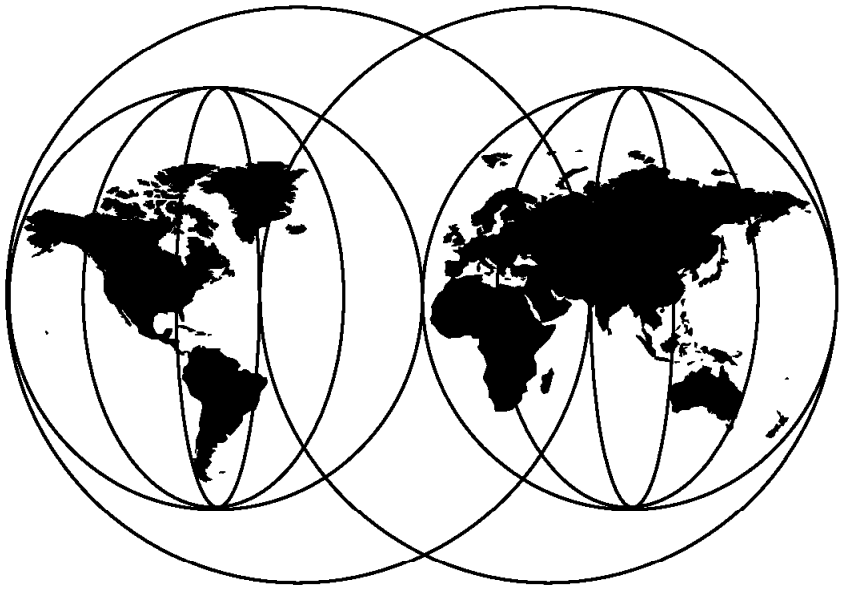
GPFS: A Parallel File System

Jason Barkes, Marcelo R. Barrios, Francis Cougard, Paul G. Crumley,
Didac Marin, Hari Reddy, Theeraphong Thitayanun



International Technical Support Organization

http://www.redbooks.ibm.com

This book was printed at 240 dpi (dots per inch). The final production redbook with the RED cover will be printed at 1200 dpi and will provide superior graphics resolution. Please see "How to Get ITSO Redbooks" at the back of this book for ordering instructions.

SG24-5165-00

SG24-5165-00

International Technical Support Organization

GPFS: A Parallel File System

April 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, “Special Notices” on page 219.

First Edition (April 1998)

This edition applies to Version 1 Release 1 of General Paralel File System for AIX (5765-B95) running Version 2, Release 4 of IBM Parallel System Support Programs for AIX (5765-529), on the IBM RS/6000 SP, for use with AIX Version 4.2.1 or 4.3.1.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HYJ Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Contents

Figures

Tables

Preface

General Parallel File System for AIX (GPFS) provides the first implementation of a truly standard parallel file system for the RS/6000 SP.

This redbook describes the installation, configuration, and use of GPFS on the RS/6000 SP. It provides examples and step-by-step instructions for installing and configuring GPFS Version 1.1.

A set of common applications and services, such as Lotus Notes Domino Server, MPI Applications, and NFS Servers, have been tested in this environment, and the results have been included in the book.

The book was written for people who are looking for guidance and advice on how to implement GPFS and how to choose the best configuration for each particular implementation. Those looking for a deeper understanding of GPFS will also find this book very helpful and detailed.

Although the book is not oriented to providing procedures for problem determination on GPFS, it contains several hints, tips and workarounds for common problems or misconfigurations.

After a brief overview and some discussion about the benefits of using GPFS as the file system of choice for the RS/6000 SP, the book unfolds all the details about this new product, including migration and compatibility issues.

A parallel file system for the RS/6000 SP has arrived, and this book coveres it all. Enjoy!

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Marcelo R. Barrios is an Advisory International Technical Support Organization (ITSO) Specialist for RS/6000 SP at the Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of RS/6000 SP. Before joining the ITSO, Marcelo worked as an assistant professor in the Electronics Department of Santa Maria University, Valparaiso, Chile. In 1993, he joined IBM as a Marketing Specialist in the RS/6000 Unit, IBM Chile.

Didac Marin is an IBM-certified AIX Advanced Technical Expert. He has a degree in Computer Science in Facultat d'Informatica de Barcelona, at the Polytechnic University of Catalunya. During the third and the fourth years of University, he worked in Hewlett-Packard with HP-UX and MPE XL in the PSO department in Barcelona. For the last three and a half years, Didac has worked in ISASA (Integracion de Sistemas Abiertos S.A.) with AIX, PSSP and everything related to these environments. He enjoys mountain hiking in Spain with his wife.

Francis Cougard is graduate of the ENSIMAG (Grenoble, France) engineering school. He has been employed by IBM for 12 years. He worked as an application developer in MVS/DB2 systems in 1985-1986, then as a presales consultant in midrange systems from 1987-1992. Francis has been an AIX I/T services specialist in PSS since 1993, and a SP Systems I/T services specialist since 1997

Jason Barkes is a certified AIX systems specialist in the AIX Systems Support Center in IBM UK Ltd. He has been employed by IBM for 18 months and has concentrated on AIX and RS/6000. Jason's previous experience includes seven years with ICL in the UK where he did a range of jobs from Customer Engineering, Junior Management to UNIX Systems Administration and Consultancy on System V and AT+T ES+. He and his wife enjoy sailing.

Theeraphong Thitayanun is an Advisory IT Specialist, IBM Thailand. He joined IBM in October 1988 as a Systems Engineer providing second level support to all VM customers in Thailand. He later transferred to an AIX unit and is responsible for providing services and supports in all areas of RS/6000 SP since Oct, 1995. He holds a degree in Computer Engineering from Chulalongkorn University and, as a Monbusho student, a masters degree in Information Technology from Nagoya Institute of Technology, Japan.

Hari Reddy is an Advisory Market Support Representative in the RS/6000 System Technologies group at Dallas System Center. Hari has been with IBM for 7 years, and he has worked in parallel computing for 10 years. Hari also conducts parallel programming workshops and consults in parallel application development.

Paul G. Crumley is a Senior Programmer in IBM's Research Division at the Center for Scalable Computing Solutions, the group that developed many parts of the SP. Before joining IBM, Paul worked at Transarc, the developer of DFS and Encina. Previous to that, he led various research projects at the Information Technology Center at Carnegie Mellon University. The ITC developed a number of distributed systems, including the AFS file system,

the ATK multimedia GUI system, and AMS, a multimedia messaging system. Paul also worked on early designs of Shark.

Thanks to the following people for their invaluable contributions to this project:

Lyle Gayne
Bob Curran
David Shapiro
Radha Kandadai
Michael Schouten
Gili Mendel
Kalyan Gunda
IBM PPS Lab Poughkeepsie

Daniel Mcnabb
Jim Wyllie
IBM Parallel File System Almaden

Robert Walkup
IBM Watson Research

Mike Buick
IBM Dallas System Center

Peter Sohn
IBM RS/6000 SP ISV Integration Poughkeepsie

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 233 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com/`
  For IBM Intranet users `http://w3.itso.ibm.com/`

- Send us a note at the following address:

  `redbook@us.ibm.com`

Chapter 1. Introduction

General Parallel File System for AIX (GPFS) provides global access to files in the RS/6000 SP. Files created in GPFS can be accessed from every node that runs GPFS code. For those nodes not running GPFS, files can still be accessed by using Network File System (NFS).

1.1 Why GPFS?

Whenever a new product comes out, there is a question that comes along with it. Should I buy and install this new product?

The answer to this question, of course, will not be the same for everybody, even with a product like a file system, which seems so general and widely usable. However, before you start thinking about this dilemma, you should consider the following points.

GPFS is part of a wider offering for the RS/6000 family. This offering consists of a set of multimedia products, where the distributed or parallel file system is one of the mechanisms used to deliver data in real time.

Data streaming was the target for the design of Tiger Shark, the ancestor of GPFS. Since then, many modifications have been made to the file system structure to integrate this distributed file system into the parallel environment of the RS/6000 SP.

A parallel file system not only offers performance advantages by eliminating the limitation of a single server for file services, but also offers a great deal of flexibility.

With a parallel file system, since all nodes can “see” the file system, it is easier to move applications from one node to another. This is especially valid for high availability solutions, where the application is moved if unrecoverable errors occur in the main server.

Sharing the same file system among several nodes has the benefit of increasing the maximum I/O bandwidth that otherwise would be limited by the maximum local I/O bandwidth of the single server.

1.2 GPFS Overview

GPFS is implemented as a standard AIX Virtual File System, which means that applications using standard AIX VFS calls (such as JFS calls) will run on top of GPFS without modifications. It also provides a byte-range locking

mechanism, allowing parallel applications to access non-overlapping blocks of a file with minimal contention.

Although GPFS is a POSIX-compliant file system, some exceptions apply to this:

- Memory mapped files are not supported in this release.
- The *stat()* is not fully supported. *mtime, atime* and *ctime* returned from the *stat()* system call may be updated slowly if the file has recently been updated on another node.

For more details, refer to *General Parallel File System for AIX: Installation and Administration*, SA22-7278.

The file system offers high scalability and high availability by allowing multiple servers and multiple disks to serve the same file system. If a server fails, the file system will still be available as long as another server has access to the disks containing the data and a network path to the client. If a disk fails, GPFS will continue providing access to the file system as long as the data contained in the failed disk is not file system metadata but user data, or as long as it has been replicated. User and file system data can be replicated or mirrored to provide an even more highly available environment.

The implementation of GPFS on the RS/6000 SP is based on two key components: the PSSP High Availability Infrastructure, and the Virtual Shared Disks. Figure 1 on page 3 shows the relationship between AIX Logical Volume Manager (LVM), PSSP, Virtual Shared Disk (VSD) subsystem, Recoverable Virtual Shared Disk (RVSD) subsystem, and GPFS.

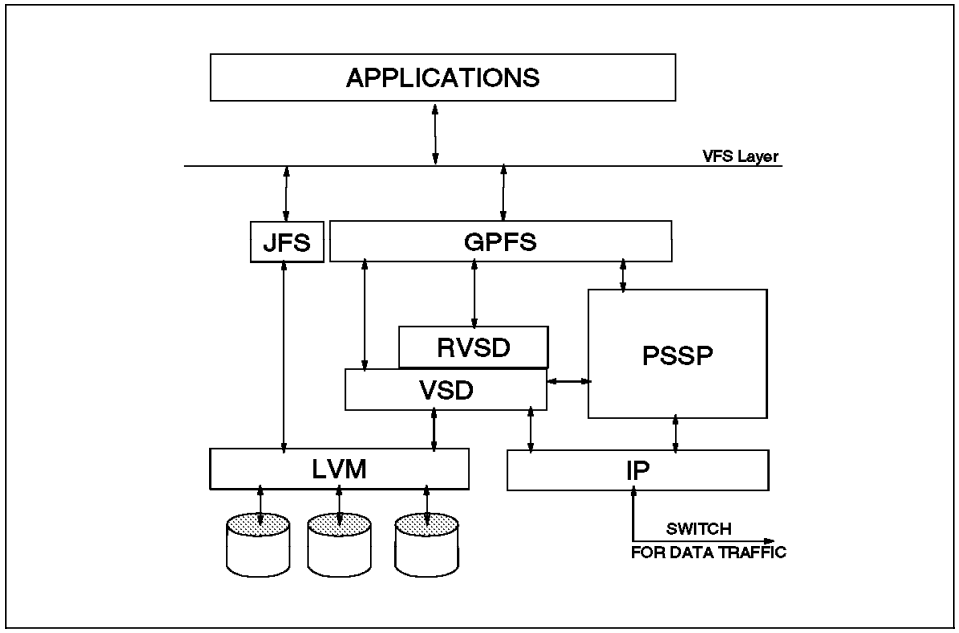


Figure 1. GPFS Overview

Each node running GPFS needs access to the data disks which comprise the local file system. GPFS uses VSD/RVSD to accomplish this and to provide a highly available environment. RVSD enables RVSD daemons to take over external disks that are attached directly as part of a loop (for SSA disks) or twin-tailed (for SCSI or serial disks).

Applications using standard system calls to manipulate files can take immediate advantage of GPFS. GPFS spreads the data, as equal as possible, across multiple servers. For more information about migration and application portability issues, refer to Chapter 4, “Migration” on page 75.

1.2.1 Implementation

GPFS is implemented as kernel extensions, a multi-threaded daemon, and a number of commands. The kernel extensions are needed to implement the Virtual File System layer that presents a GPFS file system to applications as a local file system. They are also needed to implement the Token Manager which will provide the locking mechanism.

The multi-threaded daemon provides specific functions within GPFS. Basically, the daemon provides data and metadata management (such as disk space allocation, data access, and disk I/O operations). It also provides security and quotas management.

The GPFS daemon runs on every node participating in the GPFS domain, and may take on different *personalities*. Since GPFS is not the client-server type of file system, as NFS or AFS may be seen, it uses the concept of VSD servers, which are nodes physically connected to the disks. Each node running GPFS (including VSD servers) will use the Virtual Shared Disk extensions to access the data disks.

GPFS will work within a system partition, and the nodes in this partition will be able to access the file systems from everywhere. In order to access the file systems created in GPFS, nodes need to *mount* them, like any other file system. To mount the file system, nodes have two options:

- **Nodes running GPFS**

  For these nodes, mounting a GPFS file system is the same as mounting any JFS file system. The mounting has no syntax difference with the local mounting done with JFS. At creation time, GPFS file systems can be set to be mounted automatically when nodes start up.

- **Nodes not running GPFS**

  For these nodes, GPFS file systems are made available through NFS. Nodes can mount GPFS file systems the same way they mount any NFS file system. To specify a GPFS file system and a server, these nodes must point to any node running GPFS. The GPFS file system must be exported to these nodes, just like any other local file system made available via NFS.

  Note: This applies not only to nodes that are not running GPFS, but to any NFS-capable machine.

Figure 2 on page 5 illustrates these two access methods for mounting GPFS file systems.

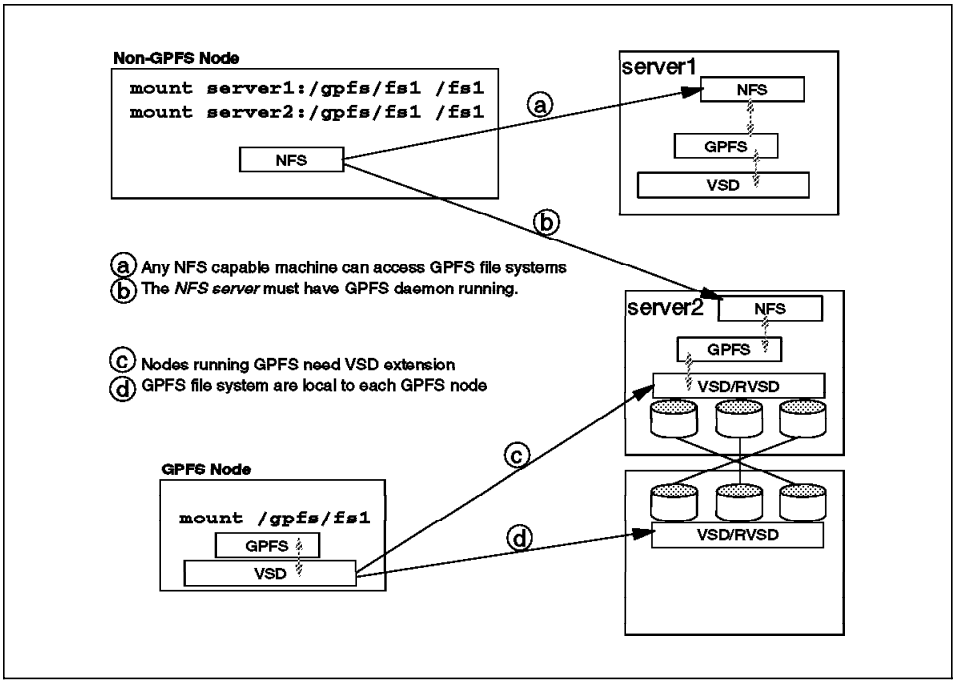


Figure 2. Mounting GPFS File Systems

GPFS lets you add/delete nodes and disks from the GPFS domain, which allows you to scale the file system to fit your needs in terms of processing or I/O bandwidth.

Because GPFS file systems can be NFS-exported, GPFS offers an alternative to NFS servers by providing better performance and reliability.

1.2.2 The GPFS Daemon

The GPFS daemon (called mmfsd) runs on every node that is part of the GPFS domain or pool. This daemon provides several services to the local node as well as to remote nodes.

The daemon provides the following local services:

- Allocation of disk space to new files.
- Management of file system metadata (manipulation of i-nodes and indirect blocks).
- Allocation of appropriate locks for data and metadata management.
- Initiation of disk I/O.
- Implementation of security and quotas.

However, all these tasks must be carried out in a parallel environment. In order to keep the consistency and the single view of GPFS file systems, the daemon implements several additional tasks. The daemon implements these tasks by assuming personalities.

A single GPFS daemon can assume more than one personality, as follows:

- Configuration Manager

  One GPFS daemon in the pool will take this personality, and it is usually the first daemon to start in the pool (the oldest). The Configuration Manager is responsible for configuration tasks within the GPFS pool, such as selecting the Stripe Group Manager for each file system.

- Stripe Group Manager

  One GPFS daemon per file system will take this personality. The Configuration Manager decides which node will act as Stripe Group Manager for each file system. It processes changes to the state of the file system, including:

  - Adding disks
  - Changing disk availability
  - Repairing the file system

  Mount and unmount processing is performed on both the Stripe Group Manager and the node requesting the service.

  The Stripe Group Manager also controls the allocations, allowing effective use of disk space and throughput.

  Finally, the Stripe Group Manager handles the token requests for file access, sending them to the Token Manager Server for the specific mount point.

- Token Manager Server

  There is one Token Manager Server per file system, located at the Stripe Group Manager node. If the Stripe Group Manager is moved to another node, the Token Manager Server moves with it. The status of the token is held in two places: on the Token Manager Server and on the Token Manager (on the requesting node) holding the token.

  In each GPFS node, there is a Token Manager, which is a kernel extension. When an application requires access to a file, it must get a token. In this situation, there are two possible scenarios:

  - The file is local (the file is served by this node).

    In this case, the application requests access to the file from the Token Manager. The Token Manager checks if this file is already granted to another application, either local or remote. If not, the

Token Manager gets a lock and passes it to the requesting application, and then informs the Token Manager Server about the request. If the file has been already granted to other nodes, the Token Manager will negotiate with those nodes in order to get the requested token.

– The file is remote (the file is served by another node).

  The Token Manager contacts the Token Manager Server for the file system (always in the same node as the Stripe Group Manager) to request a token. If other nodes already have the token for this file, the Token Manager Server will send back the list of nodes having the token. It is the responsibility of the requesting Token Manager to negotiate with the nodes in that list to obtain a token.

- Metadata Manager

  One GPFS daemon per file will take this personality. A GPFS daemon may be the Metadata Manager of multiple files. In almost all cases, the node that has had the file open for the longest period of continuous time is the Metadata Manager.

Note: All these personalities may be in the same GPFS daemon. However, for performance reasons, it is advisable to distribute the load of the different tasks across the pool of nodes. For more details about what happens if one or several of these services fail within GPFS, refer to Chapter 3, “Failover Scenarios” on page 57.

1.2.3 Allocation Mechanism

GPFS uses a standard i-node structure for file allocation. An *i-node* is a structure which contains file information. Each file has at least one i-node, and each i-node contains information for only one file.

A GPFS file system consists of a set of disks distributed across the RS/6000 SP and served by VSD servers. When the file system gets created, several structures are put in place to keep the information about this file system. The following are the structures needed and created by GPFS for each file system:

- i-node

  Every file on a disk has an i-node. The i-node contains attributes of the file (size, owner, permissions, and so on), plus the information necessary to locate the data blocks of the file.

- i-node allocation file

  The i-node allocation file represents the availability of i-nodes for the creation of new files in the file system. This cannot be changed after the

file system is created, so careful planning is needed in order to create enough i-nodes for all the files expected to be created in the file system.

- Block allocation file

  The block allocation file is a collection of bits that represent the availability of disk space within the stripe group. One unit in the allocation map represents a sub-block (1/32 of the block size) of the file system.

- Log files

  Log files contain all the transactions executed against the file system. Additional log files may be created if needed. Log files are always replicated. There are two copies of the log for each executing node.

When a file is created, an i-node is allocated. This i-node will contain the following (along with other information):

- The owner's ID
- The owner's GID
- Permissions (including extended ACL)
- Date/time for creation/last access/last modification
- Number of data replicas
- Number of metadata replicas
- Data itself (for very small files, less than 12 bytes)
- Pointers to data blocks (direct blocks)
- Pointers to indirect blocks (for large files)

Figure 3 on page 9 describes the file structure in a GPFS file system.

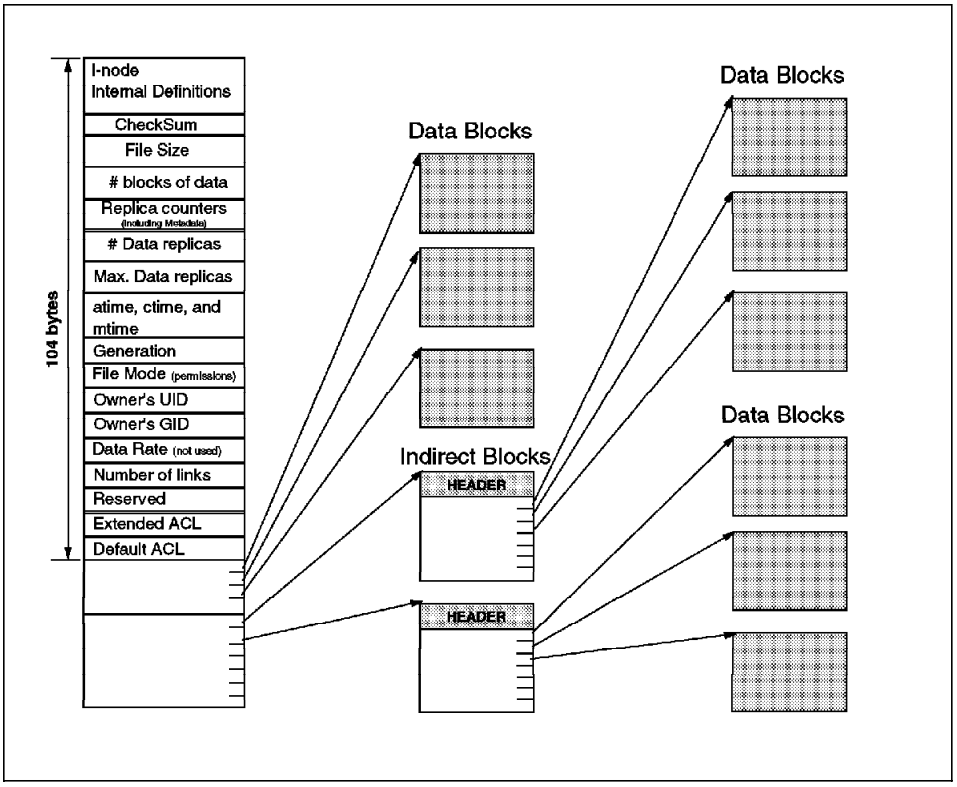


Figure 3. File Structure

The i-node size can be chosen when the file system gets created. It can be as little as 512 bytes, or as big as 4KB. The i-node size will have an impact on the maximum file size. For exact numbers, refer to Appendix B, "GPFS Maximum File Size and Related Parameters" on page 189.

If a file is bigger than:

```
Block Size * ((i-node size - 104) / (# Data Replicas * 6))
```

bytes,[1] then indirect blocks are needed. GPFS supports only one level of indirection in this release.

[1] The header of an i-node is 104 bytes in length. A disk pointer is six bytes in length.

Indirect blocks contain pointers to data blocks. Each indirect block can point to up to

```
(Indirect Block Size - 44) / (# Metadata Replicas * 6)
```

data blocks.[2]

Since all the parameters can be selected when the file system is created, the maximum file size and the maximum file system size will vary. For a list of maximum size values, refer to Appendix B, “GPFS Maximum File Size and Related Parameters” on page 189.

1.2.4 Striping

Each file system consists of a number of VSD disks. The set of disks making up a file system is called a *stripe group*. The data and metadata allocation within this stripe group is done by the Stripe Group Manager, as described earlier.

GPFS offers three different algorithms for striping data and metadata across the VSD disks.

- RoundRobin

  This is the default method. The data and metadada blocks are written one on each VSD disk, looping around the stripe group. This method offers good performance and correct balance of the traffic between the VSD disks.

- Random

  In this method, each block of data or metadata is written on a VSD disk selected randomly. This method does not assure that the traffic will be balanced among the VSD disks.

- BalancedRandom

  This method is very similar to RoundRobin, with the difference that each data or metadata block is written to a VSD disk selected randomly. However, as in the RoundRobin method, the same VSD disk is not selected until all the disks within the stripe group have been used.

The stripe method is defined per file system. Choosing a correct striping method is critical for correctly tuning specific applications.

---

[2] These 44 bytes include the indirect block header and trailer.

1.2.5 Replication

Data availability and performance are among the main concerns for any file system. GPFS environments offer several choices to accomplish these goals, including replication. As shown in the following chapters, replication offers a good alternative for providing both data availability and performance. However, like everything else, replication has a cost, and the costs are associated with:

- Maximum file size GPFS is able to handle
- Performance effects

GPFS allows up to two copies for data and for metadata. The number of data and metadata replicas can be set at file or file system level. By default, data and metadata are not replicated. Only log files are replicated, keeping two copies per node, as explained earlier.

For replication, GPFS uses the concept of a *failure group*, which defines single points of failure. It uses this concept to locate the data and metadata replicas in different failure groups, and in this way improves file system availability.

More details about replication can be found at 2.2, “Configuration” on page 30.

1.2.6 Locking Mechanism

GPFS provides a locking mechanism based on tokens. When a file system is created, one node within the pool is selected as the Token Manager Server for that file system (it is always the same node where the Stripe Group Manager resides). This Token Manager Server will handle all requests for access to a file in the file system.

Each GPFS client (nodes where the GPFS daemon is running) has a kernel extension called Token Manager, which is responsible for getting the tokens requested by local applications.

Tokens can be requested/granted for byte-range or file locking. There are two kinds of locks: GPFS internal locks, which are file/byterange, and POSIX locks (fcntl system call), which are available to applications. The POSIX locks are advisory; the internal locks are enforced.

Chapter 2. Installation and Configuration

The General Parallel File System (GPFS) environment is specific to AIX on the RS/6000 SP. Various software requirements must be installed and configured correctly before you can create a GPFS file system. Guidelines are provided in the following sections that enable you to configure the basics and get you started with GPFS.

This chapter guides you in installing the pre-requisite software products and helps you get GPFS up and running as quickly as possible. It also focuses on considerations when configuring different scenarios within your SP environment.

A background knowledge of Virtual Shared Disk and IBM Recoverable Shared Disk is advisable. However, because GPFS takes the guesswork out of the creation of Virtual Shared Disk and IBM Recoverable Virtual Shared disk by constructing them for you, you should increase your familiarity with the concepts behind these products. Further information about these packages can be found in the *IBM Parallel System Support Program for AIX: Managing Shared Disks,* SA22-7279.

2.1 Installation

The majority of the GPFS installation work is carried out on the SP nodes. However, there is a small amount of installation work that needs to be completed on the Control Workstation. There are many ways of installing the software and the general use of SMIT for installation work within the SP framework is *not* recommended. Instead, you should use the approved method that is described both here and in the *General Parallel File System for AIX: Installation and Administration Guide,* SA22-7278.

The approach documented here is a very methodical way of installing the packages. There are many ways of completing the installation and you should follow any on-site procedures already in place. The majority of the system management and configuration work cannot be performed directly at the Control Workstation. You will need to be on a GPFS node to execute most GPFS commands or alternatively you dsh/rsh from the Control Workstation to execute these commands.

2.1.1 Hardware Requirements

1. RS/6000 SP
2. HiPS or SP Switch
3. Sufficient disk capacity to support file systems and failover scenarios

Any twin-tailed disk supported by the logical volume manager running on AIX 4.2.1 is a good choice for the configuration of GPFS because the IBM Recoverable Shared Disk will be able to fully utilize its failover functionality.

2.1.2 Software Requirements

The main software requirements for the GPFS environment and a successful installation are:

- AIX 4.2.1 or 4.3.1
- Parallel System Support Program 2.4
- IBM Recoverable Virtual Shared Disk 2.1.1

The full details of these products are:

1. AIX Version 4 Release 2.1 (5765-655 or 5765-C34) or 4.3.1
2. Parallel System Support Programs for AIX, Version 2 Release 4 or later (5765-529) with the following options installed:
   - SP System Support Package (ssp.basic)
   - SP Communication Subsystem Package (ssp.css)
   - Group Services High Availability Subsystem (ssp.ha)
   - Sysctl (ssp.sysctl)
   - SP Centralized Management Interface (ssp.csd.cmi)
   - SP VSD Usability Package (ssp.csd.sysctl)
3. IBM Recoverable Virtual Shared Disk Version 2.1.1 (5765-646) or later with the following options installed:
   - SP High Availability RVSD (rcsd.rvsd)
   - SP RVSD Base Package (rcsd.vsd)

2.1.3 Installation Road Map and Checkpoints for Existing Setup

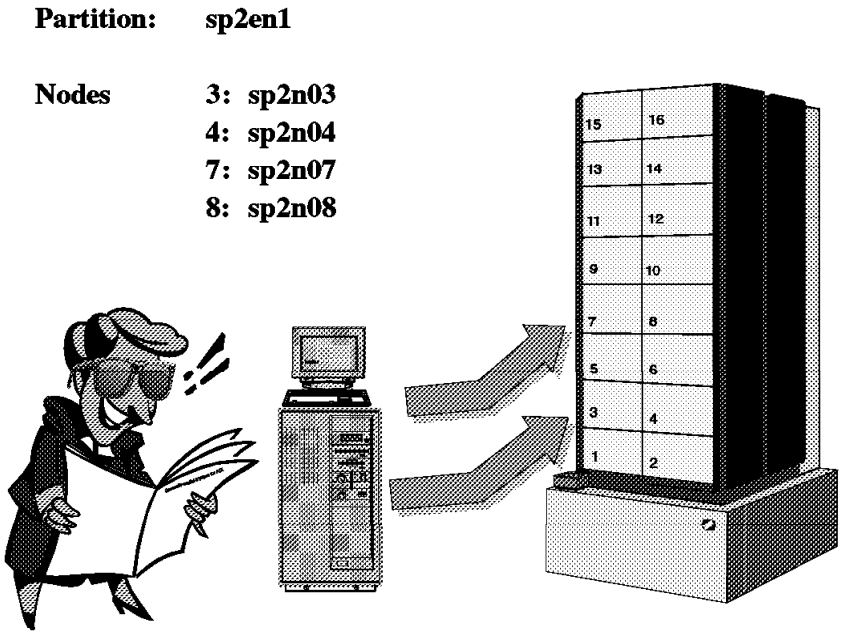
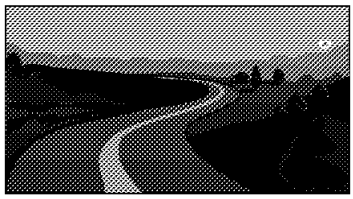


Figure 4. Partition Setup for Installation

The setup shown here is a partition of four SP thin nodes held within a frame of 16. The partition name is sp2en1 and the nodes are sp2n03, sp2n04, sp2n07, and sp2n08, respectively. The switch used in this example was a HiPS switch; however, this is not important as either type of switch is supported in this release of GPFS. This will be the base for the installation shown in this part of the book.



The roadmap shown in Table 1 shows you the paths to follow within this chapter:

Table 1. Installation Road Map

S/W Start Point	Refer to: Checkpoint:
< AIX 4.2.1 < PSSP 2.4	*IBM Parallel System Support Programs for AIX: Installation and Migration Guide,* GC23-3898 Checkpoint: **1** **2**
AIX 4.2.1/4.3.1 PSSP 2.4	2.1.3.2, “Minimal Virtual Shared Disk Installation” on page 19 2.1.3.3, “Minimal IBM Recoverable Virtual Shared Disk Installation” on page 19 2.1.4, “GPFS Software Installation” on page 20 Checkpoint: **1** **2**
AIX 4.2.1/4.3.1 PSSP 2.4 VSD 2.4	2.1.3.3, “Minimal IBM Recoverable Virtual Shared Disk Installation” on page 19 2.1.4, “GPFS Software Installation” on page 20 Checkpoint: **1** **2** **3**
AIX 4.2.1/4.3.1 PSSP 2.4 VSD 2.4 RVSD 2.1.1	2.1.4, “GPFS Software Installation” on page 20 Checkpoint: **1** **2** **3** **4**

1 Check the level of parallel system support program:

```
sp2en1> lslpp -l ssp.basic
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  ssp.basic                 2.4.0.0  COMMITTED  SP System Support Package

Path: /etc/objrepos
  ssp.basic                 2.4.0.0  COMMITTED  SP System Support Package
sp2en1> dsh lslpp -l ssp.basic
```

2 Check the level of AIX on the Control Workstation and the SP nodes:

```
sp2en1> oslevel
4.2.1.0
sp2en1> dsh oslevel
sp2n03: 4.2.1.0
sp2n04: 4.2.1.0
sp2n07: 4.2.1.0
sp2n08: 4.2.1.0
```

3 Check the level of virtual shared disk on the SP nodes:

```
sp2en1> dsh lslpp -l ssp.csd*
  Fileset                 Level    State    Description
   ----------------------------------------------------------------------
Path: /usr/lib/objrepos
  ssp.csd.hsd           2.4.0.0  COMMITTED  SP VSD Data Striping package
                                            (ssp.csd.hsd)
  ssp.csd.sysctl        2.4.0.0  COMMITTED  SP VSD USABILITY IMPROVEMENT
                                            (ssp.csd.sysctl)
  ssp.csd.vsd           2.4.0.0  COMMITTED  SP IBM Virtual Shared Disk
                                            Package (ssp.csd.vsd)

Path: /etc/objrepos
  ssp.csd.hsd           2.4.0.0  COMMITTED  SP VSD Data Striping package
                                            (ssp.csd.hsd)
  ssp.csd.sysctl        2.4.0.0  COMMITTED  SP VSD USABILITY IMPROVEMENT
                                            (ssp.csd.sysctl)
  ssp.csd.vsd           2.4.0.0  COMMITTED  SP IBM Virtual Shared Disk
                                            Package (ssp.csd.vsd)
```

Check your level of virtual shared disk on the Control Workstation:

```
sp2en1> lslpp -l ssp.csd*
  Fileset                 Level    State    Description
   ----------------------------------------------------------------------
Path: /usr/lib/objrepos
  ssp.csd.cmi           2.4.0.0  COMMITTED  SP Centralized Management
  ssp.csd.vsd           2.4.0.0  COMMITTED  SP IBM Virtual Shared Disk
                                            Package (ssp.csd.vsd)

Path: /etc/objrepos
  ssp.csd.vsd           2.4.0.0  COMMITTED  SP IBM Virtual Shared Disk
                                            Package (ssp.csd.vsd)
```

If you do not have the virtual shared disk package installed, then follow 2.1.3.2, “Minimal Virtual Shared Disk Installation” on page 19.

4 Check your level of IBM Recoverable Shared Disk on the SP nodes:

```
sp2en1> dsh lslpp -l rcsd*
  Fileset                 Level  State      Description
  ----------------------------------------------------------------------
Path: /usr/lib/objrepos
  rcsd.hahc             2.1.1.0  COMMITTED  SP HA Recoverable VSD Package
  rcsd.rvsd             2.1.1.0  COMMITTED  SP HA Recoverable VSD Package
  rcsd.vsd              2.1.1.0  COMMITTED  SP IBM Recoverable Virtual
                                            Shared Disk base VSD Package

Path: /etc/objrepos
  rcsd.hahc             2.1.1.0  COMMITTED  SP HA Recoverable VSD Package
  rcsd.rvsd             2.1.1.0  COMMITTED  SP HA Recoverable VSD Package
  rcsd.vsd              2.1.1.0  COMMITTED  SP IBM Recoverable Virtual
                                            Shared Disk base VSD Package
```

If you do not have the Virtual Shared Disk package installed, then follow 2.1.3.3, “Minimal IBM Recoverable Virtual Shared Disk Installation” on page 19.

2.1.3.1 Guidelines to Migrate Virtual Shared Disk Software

A new version of Virtual Shared Disk is shipped with PSSP 2.4 and therefore ssp.csd* will have to be migrated to the new version.

You will need to unconfigure all your existing Virtual Shared Disks before installing the new Virtual Shared Disk software. The general steps taken here are to suspend, stop and unconfigure your Virtual Shared Disks:

```
sp2en1> dsh suspendvsd -a
sp2en1> dsh stopvsd -a
sp2en1> dsh ucfgvsd -a
```

For more details about these commands and their syntax see *IBM Parallel System Support Program for AIX: Managing Shared Disks,* SA22-7279. You should make sure that they are fully unconfigured with the *dsh lsvsd -l* command. You will receive the following error from each of your nodes if your Virtual Shared Disks have been successfully unconfigured:

```
sp2en1> dsh lsvsd -l
sp2n03: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n03: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
sp2n04: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n04: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
sp2n07: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n07: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
sp2n08: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n08: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
```

The unconfiguration of the Virtual Shared disks has to be completed before you install the Parallel Systems Support Program Version 2 Release 4. See *IBM Parallel System Support Programs for AIX: Installation and Migration Guide,* GC23-3898 for further information.

2.1.3.2 Minimal Virtual Shared Disk Installation

The following steps show you a method to install the prerequisite Virtual Shared Disk package that is required to run GPFS. If you are installing your SP system from fresh, you may wish to use the */etc/script.cust* file to install your SP nodes.

To perform this installation:

1. Create a subdirectory on the Control Workstation with the following command:

   ```
   mkdir /spdata/sys1/install/pssplpp/PSSP-2.4/vsd
   ```

2. Use *smit bffcreate* to copy the Virtual Shared Disk software to hard disk for future installation at the preceding directory. Select all filesets pertaining to ssp.csd.

3. Install these filesets at the Control Workstation as follows:

   ```
   sp2en1> cd /spdata/sys1/install/pssplpp/PSSP-2.4/vsd
   sp2en1> installp -agXd ″.″ ssp.csd.cmi ssp.csd.vsd
   ```

4. Install all ssp.csd filesets to the nodes that will run GPFS, or Virtual Shared Disk, or both.

   - NFS-export, and then mount the file system to all nodes as follows:

     ```
     sp2en1> /usr/sbin/mknfsexp -d /spdata/sys1/install/pssplpp/PSSP-2.4 /
             -t ro -B
     sp2en1> dsh mount sp2en1:/spdata/sys1/install/pssplpp/PSSP-2.4
     ```

   - Run the installp command on all nodes:

     ```
     sp2en1> dsh installp -agXd /mnt/vsd ssp.csd.hsd ssp.csd.sysctl /
             ssp.csd.vsd
     ```

You can now verify your Virtual Shared Disk installation by using `lslpp -l ssp.csd*` against the former output.

2.1.3.3 Minimal IBM Recoverable Virtual Shared Disk Installation

Now you must install the IBM Recoverable Virtual Shared Disk software on your system. To do this, you can use a procedure similar to that described in 2.1.3.2, “Minimal Virtual Shared Disk Installation.”

This product is a separately Licensed Program Product (LPP); it is not part of the Parallel Systems Support Program. The main differences in this procedure are that the only installation work that needs to be carried out is on your SP nodes and that you will not have to export the directory (it will already be exported).

The IBM Recoverable Virtual Shared Disk package will have to be installed on all SP nodes that either contain Virtual Shared Disks as a server or, that will run the GPFS file system daemon even if you are not running twin-tailed disks. The reason for this is that the IBM Recoverable Shared Disk software makes use of the `fencevsd` and `unfencevsd` facilities now available in this release.

To perform this installation:

1. Check that /spdata/sys1/install/pssplpp/PSSP-2.4 is mounted over /mnt. If this is not the case, then follow the first part of step 4 in 2.1.3.2, "Minimal Virtual Shared Disk Installation" on page 19.
2. Create a subdirectory on the Control Workstation with the following command:

   ```
   sp2en1> mkdir /spdata/sys1/install/pssplpp/PSSP-2.4/rvsd
   ```

3. Use `smit bffcreate` to copy IBM Recoverable Virtual Shared Disk software to hard disk for future installation at the preceding directory. Select all filesets pertaining to rcsd.
4. Install all rcsd filesets to the nodes that will run on GPFS, or Virtual Shared Disk, or both:

   ```
   sp2en1> dsh installp -agXd /mnt/rvsd all
   ```

You can now verify your IBM Recoverable Virtual Shared Disk installation by using the `lslpp -l rcsd*` command against the former output.

2.1.4 GPFS Software Installation

The installation of the GPFS software follows the same procedure as the previous packages documented here.

To perform this installation:

1. Check that /spdata/sys1/install/pssplpp/PSSP-2.4 is mounted over /mnt. If this is not the case, then follow the first part of step 4 in 2.1.3.2, "Minimal Virtual Shared Disk Installation" on page 19.
2. Create a subdirectory on the Control Workstation with the following command:

   ```
   sp2en1> mkdir /spdata/sys1/install/pssplpp/PSSP-2.4/mmfs
   ```

3. Use `smit bffcreate` to copy the GPFS software to hard disk for future installation at the preceding directory. Select all filesets pertaining to mmfs.
4. Install the single fileset required on the Control Workstation as follows:

   ```
   sp2en1> installp -agXd /spdata/sys1/install/pssplpp/PSSP-2.4/mmfs mmfs.gpfs
   ```

   The GPFS daemon does not run on the Control Workstation. Therefore, only part of the GPFS code needs to be loaded.
5. Install all mmfs filesets to the nodes that will run the GPFS daemon. If you intend to install SP nodes as devoted Virtual Shared Disk servers, you will not have to load the package onto these nodes. You may wish to create a working collection variable *WCOLL* to match your GPFS nodes. Alternatively, you can specify the individual commands from the dsh command, as follows:

   ```
   sp2en1> dsh -w sp2n03,sp2n04,sp2n07,sp2n08 installp -agXd /mnt/mmfs all
   ```

You can now use the `lslpp -l` command to verify the installation of the GPFS filesets on each of your nodes:

```
sp2en1> dsh -w sp2n03 lslpp -l mmfs*
  Fileset                  Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  mmfs.base.cmds         3.0.0.0  COMMITTED  Multimedia File Manager
                                             Commands
  mmfs.base.rte          3.0.0.0  COMMITTED  Multimedia File Manager
  mmfs.gpfs.rte          1.1.0.0  COMMITTED  GPFS File Manager
  mmfs.msg.en_US         3.0.0.0  COMMITTED  Multimedia Server Messages -
                                             U.S. English
  mmfs.util.cmds         3.0.0.0  COMMITTED  Multimedia Server Utilities
  mmfs.util.smit         3.0.0.0  COMMITTED  Multimedia Server SMIT Panel

Path: /etc/objrepos
  mmfs.base.rte          3.0.0.0  COMMITTED  Multimedia File Manager
  mmfs.gpfs.rte          1.1.0.0  COMMITTED  GPFS File Manager

Path: /usr/share/lib/objrepos
  mmfs.man.en_US.data    3.0.0.0  COMMITTED  Multimedia Server Man Pages-
                                             U.S. English
```

Note: At this point you should make sure that the path on your Control Workstation and your SP nodes includes the following line:

- `/usr/lpp/csd/bin`

Also insure that this path is set only on your SP nodes:

- `/usr/lpp/mmfs/bin`

2.1.5 Authorization for Kerberos and Sysctl

Since Virtual Shared Disk and GPFS use sysctl commands, you have to update the relevant sysctl ACL files. The following screen outputs show you the root.admin default setup for sysctl. The files must be edited on the Control Workstation, as well as on the SP nodes. The three files that must be edited are: /etc/sysctl, /etc/sysctl.vsd.acl, and /etc/sysctl.mmcmd.acl:

- The screen output from /etc/sysctl.acl:

```
sp2en1> cat /etc/sysctl.acl
#acl#
# This sample acl file contains commented out lines for a principal
# and an acl file.

_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
```

- The screen output from /etc/sysctl.vsd.acl:

```
sp2en1> cat /etc/sysctl.vsd.acl
#acl#

# These are the users that can issue sysctl_vsdXXX command on this node
# Please check your security administrator to fill in correct realm name
# you may find realm name from /etc/krb.conf

_PRINCIPAL root@MSC.ITSO.IBM.COM
_PRINCIPAL rcmd@MSC.ITSO.IBM.COM
```

- The screen output from /etc/sysctl.mmcmd.acl:

```
sp2en1> cat /etc/sysctl.mmcmd.acl
#acl#
# These are the users that can issue multinode mmfs commands through sysctl:
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
_PRINCIPAL root.@MSC.ITSO.IBM.COM
_PRINCIPAL rcmd.@MSC.ITSO.IBM.COM
```

Note: The " . " after the user shown here is important for this installation.

Once you have updated these files, you must refresh sysctl to complete the database update for the new entries as follows:

```
sp2en1> dsh sysctl svcrestart
```

If sysctl svcrestart fails with the following message:

```
sysctl:  2501-122 svcrestart: Insufficient Authorization.
```

then it is likely that your files are incorrectly edited or have not been updated on all the SP nodes and the Control Workstation.

To check your sysctl authorization, first run klist to look at your ticket, and then run sysctl whoami. To check that you can run the IBM Virtual Shared Disk commands, issue sysctl sysctl_vsdcheck. You will also need to make sure that your multi-node commands function correctly. You can do this by running vsdsklst -a. This command will list information about physical and logical volume manager states as a view for the Virtual Shared Disk software.

The following shows sample output on a SP node for vsdsklst -n:

```
sp2en1> vsdsklst -n 8
Node Number:8; Node Name:sp2n08.msc.itso.ibm.com
    Volume group:rootvg; Partition Size:4; Total:250; Free:33
        Physical Disk:hdisk0; Total:250; Free:33
    Not allocated physical disks:
        Physical disk:hdisk1; Total:1.0
        Physical disk:hdisk2; Total:2.0GB
        Physical disk:hdisk3; Total:2.0GB
        Physical disk:hdisk4; Total:2.0GB
```

You can check that GPFS commands are available to use from sysctl as follows:

```
sp2en1> dsh sysctl mmremote single
sp2n03: ok
sp2n04: ok
sp2n07: ok
sp2n08: ok
```

Again, if any of these commands fail, you need to check your ACL files. You should also check file /etc/sysctl.conf and make sure the following lines are included:

```
sp2en1> tail -4 /etc/sysctl.conf
# Include VSD sysctl_vsd.cmd commands
include /usr/lpp/csd/sysctl/sysctl_vsd.cmds
# Include mmfs sysctl commands
include /usr/lpp/mmfs/bin/mmcmdsysctl
```

In addition, you may need to run a `kinit` on all SP nodes and the Control Workstation for *root.admin*.

Sysctl can also be restarted by issuing:

```
sp2en1> dsh stopsrc -s sysctl
sp2en1> dsh startsrc -s sysctl
```

You should also stop and restart this daemon on your Control Workstation. Refer to *Parallel System Support Guide* or *IBM RS/6000 SP Management, Easy, Lean and Mean,* GG24-2563 for further information about kerberos and sysctl.

2.1.6 Tuning and Verifying the Switch

If you already have Virtual Shared Disk software installed, you must check the *rpoolsize* and *spoolsize* values on each node. The value for both of these variables is 16777216. These values cannot be set on the Control Workstation. To check these values run the following:

```
sp2en1> dsh -w sp2n03 lsattr -El css0
sp2n03: bus_mem_addr   0x04000000 Bus memory address        False
sp2n03: int_level      0xb        Bus interrupt level       False
sp2n03: int_priority   3          Interrupt priority        False
sp2n03: dma_lvl        9          DMA arbitration level     False
sp2n03: spoolsize      16777216   Size of IP send buffer    True
sp2n03: rpoolsize      16777216   Size of IP receive buffer True
sp2n03: adapter_status css_ready  Configuration status      False
```

If these values are incorrect, you need to change them to 16777216.

If this is a fresh installation, you must also check the level of paging space available to the system. If you have not customized your system, it is likely that you will only have a small amount of paging. You must increase the paging because you are about to pin 32 megabytes of memory. You can change your value by running:

```
dsh /usr/lpp/ssp/css/chgcss -l css0 -a rpoolsize=16777216 -a spoolsize=16777216
```

You can ignore the following messages, which will be returned from each SP node:

```
sp2n03: There was customized rpoolsize, and new value != default
sp2n03: There was customized spoolsize, and new value != default
```

At this point you can follow one of two paths to make your new switch values active, they are not dynamic. You can either reboot your system or you can remove and reconfigure your switch adapter on each of your SP nodes (as shown on SP node sp2n03):

```
sp2n03> Efence 3
sp2n03> cd /usr/lpp/ssp/css
sp2n03> ./ifconfig css0 down
sp2n03> ./ifconfig css0 detach
sp2n03> ./rc.switch
sp2n03> Eunfence 3
```

If you are running a HiPS switch, then you will probably need to run an `Estart` depending on the Eduration set on your system.

2.1.7 Tuning and Verifying Virtual Shared Disk

At this point in the installation, you are required to enter the disk parameters for each of your SP nodes into the System Data Repository (SDR). These parameters must be set in order for GPFS to operate efficiently. These values are set only on the Control Workstation:

```
vsdnode 3 4 7 8 css0 64 256 256 48 4096 262144 33 61440
```

You can verify the level of these values by using `vsdatalst -n`. Also, if you are already running a Virtual Shared Disk system, you can update these values with the Virtual Shared Disk command `updatevsdnode`. For example, if you need to change the value of your *max_buddy_buffers* from 16 to the recommended 33 on all your SP nodes, you must run:

```
sp2en1> updatevsdnode -n ALL -s 33
```

The syntax and values for these commands are discussed in greater detail in *Parallel System Support Program for AIX: Managing Shared Disks,* SA22-7279.

2.1.8 Deleting Old Rollback Files

If you issue a Virtual Shared Disk command that operates on multiple SP nodes (such as `createvsd`), or a GPFS command (such as `mmcrfs`) that updates the VSD configuration, and the command fails, a rollback file will be created so that a second invocation of that command can resume the last successful operation. (A command that is issued against multiple SP nodes fails if any of the nodes cannot execute it). If you later change your configuration or run a different command, it tries to complete processing using the rollback file and will fail.

To be sure that here are no rollback files from failed invocations of your commands, you can delete them using the `rm` command. The file is found in /usr/lpp/csd/vsdfiles and is called *vsd_rollback.*

Any time a command of this nature fails, you need to check for old rollback files.

2.1.9 Creation of Single Virtual Shared Disk for New Installation

The GPFS relies on several instances before it will start. The instances include the availability of High Availability services, the switch, and the IBM Recoverable Virtual Shared Disk daemon.

The last IBM Recoverable Virtual Shared Disk daemon cannot start unless you have at least one Virtual Shared Disk available to it; therefore, you must create one manually. You can achieve this through either SMIT, Perspectives, or from the command line as shown here.

This Virtual Shared Disk does not have to be part of your GPFS configuration. It can sit on any volume group that has a spare logical partition. The example shown here is for the creation of a Virtual Shared Disk in *rootvg*. The work has to be carried out from the Control Workstation and creates the Virtual Shared Disk on SP node sp2n03:

```
sp2en1> createvsd -n 3/:hdisk0/ -g rootvg -s 1
```

You can then see your Virtual Shared Disk by running:

```
sp2en1> vsdatalst -v
```

The -v option shows the Virtual Shared Disk SDR information and looks similar to:

```
                VSD Table
VSD name   Logical volume  Global Volume Group  minor# option  size_in_MB
-------------------------  -------------------  ------ ------  ----------
vsd1n3     lvvsd1n3        rootvgn3             5      nocache      1
```

You must now configure and start your new Virtual Shared Disk on the SP nodes as follows:

```
dsh cfgvsd vsd1n3
dsh startvsd vsd1n3
```

Check that your Virtual Shared Disk has started correctly:

```
sp2en1> dsh -w sp2n03 lsvsd -l
sp2n03: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n03:  5      ACT    3      10      12      vsd1n3     nocache 1
```

You can see from the *state* field that the Virtual Shared Disk is active. You should check that this is the case for all your SP nodes. The *lv_major* and *lv_minor* values will be zero on the other nodes since the Vitual Shared Disk does not reside there.

You can now verify that your software has created the Virtual Shared Disk correctly. You can do this by running the vsdvts test suite. This has to run on a SP node because it is interactive:

```
sp2n03> vsdvts vsd1n3
NOTE:  This command will change the content of vsd1n3 !!!
       Do you want to continue y/n? y
vsdvts: Step 1: Writing file /unix to VSD vsd1n3.
dd if=/unix of=/dev/rvsd1n3 count=256 bs=4096 seek=1
256+0 records in.
256+0 records out.
vsdvts: Step 1 Successful!
vsdvts: Step 2: Reading data back from the VSD.
dd of=/tmp/vsdvts.4550 if=/dev/rvsd1n3 count=256 bs=4096 skip=1
256+0 records in.
256+0 records out.
vsdvts: Step 2 Successful!
vsdvts: Step 3: Verifying data read from the VSD.
dd if=/unix count=256 bs=4096 ] cmp -s - /tmp/vsdvts.4550
256+0 records in.
256+0 records out.
vsdvts: Step 3 Successful!

VSD Verification Test Suite Successful!
```

Now that you have successfully verified your Virtual Shared Disk, you will have to suspend, stop and unconfigure the Virtual Shared Disk before continuing with the configuration of GPFS:

```
dsh suspendvsd vsd1n3
dsh stopvsd vsd1n3
dsh ucfgvsd vsd1n3
```

Leave a few minutes between the preceding steps to allow Virtual Shared Disk software to do its work. Then check that the Virtual Shared Disk has been unconfigured. If your Virtual Shared Disks have been successfully unconfigured, you will receive the following error from each of your SP nodes:

```
sp2en1> dsh lsvsd -l
sp2n03: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n03: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
sp2n04: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n04: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
sp2n07: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n07: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
sp2n08: minor  state server lv_major lv_minor vsd-name   option  size(MB)
sp2n08: lsvsd: 0034-002 Error opening vsd /dev/VSD0.
```

The Virtual Shared Disk can, of course, be part of your setup. For experienced users with existing Virtual Shared Disk and IBM Recoverable Shared Disk, this may be preferable.

Figure 5 shows the transitional states, commands, and types of operations that can be performed at each level with Virtual Shared Disk:

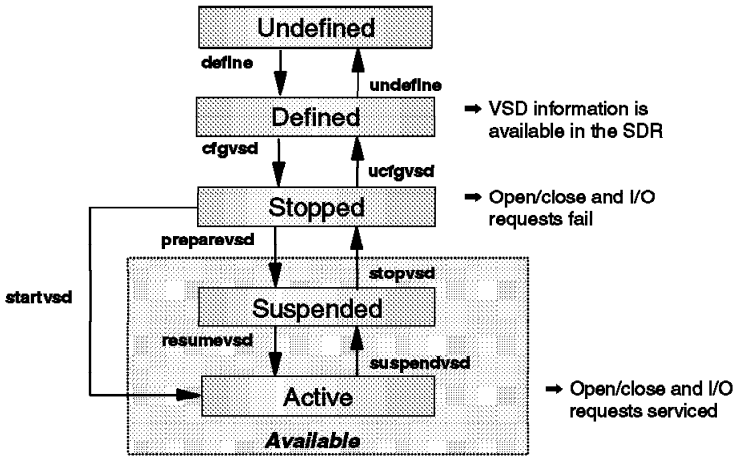


Figure 5. VSD State Transitions

2.1.10 High Availability Heartbeat Sensitivity

The GPFS daemon utilizes SP high availability services. This increases the amount of work it has to complete to stabilize the group. The system default allows for four missed heartbeats in its normal operation. We recommend that you increase this value from four missed heartbeats to six because of this increased workload. This increase may still not be enough if you are using a mixture of nodes with different CPU performance and load. To check the value, run the following on the Control Workstation;

```
sp2en1> SDRGetObjects TS_Config
Frequency    Sensitivity  Run_FixPri   FixPri_Value Log_Length
        1              4           1             38       5000
```

To change this value, you can run:

```
sp2en1> SDRChangeAttrValues TS_Config Sensitivity==6

sp2en1> SDRGetObjects TS_Config
Frequency    Sensitivity  Run_FixPri   FixPri_Value Log_Length
        1              6            1             38       5000
sp2en1> dsh refresh -s hats
sp2en1> refresh -s sp2en1.hats
```

You must then refresh Topology Services.

2.1.11 The lsof Command (List Open Files)

The lsof utility is very useful in diagnosing unmounting problems not just within GPFS, but in various different file systems. We recommend that you install this command on any node running GPFS.

You can acquire a pre-built binary for AIX from:

`aixpdslib.seas.ucla.edu`

You can get the latest version via anonymous ftp from:

`vic.cc.purdue.edu (cd to /pub/tools/unix/lsof)`

The inventor of the `lsof` command is Abell (abe@cc.purdue.edu), Purdue University Computing Center. For more information about this command, refer to F.3, “LiSt Open File” on page 213.

2.2 Configuration

This section examines the planning and configuration needed to create and run a GPFS file system. There are a number of factors that cannot be changed at a later date once your file system has been created. If these decisions are incorrect and have a critical impact on your business, you will have to take the standard UNIX approach to this problem by archiving and restoring your data. Examples of these “no return” decisions are i-node size, indirect block size, and the estimated number of nodes to which the file system will be mounted.

However, many of the decisions are dynamic and can be changed once your file system is up and running. Examples of this are default replication, file system automount, and nodes that are available to the file system.

2.2.1 Considerations

Following are the various factors that you must consider when planning your configuration, including how best to optimize your resources. Table 2 shows the factors that are either tunable or static once the file system has been created, and indicates where restripe should be considered.

Table 2. Tunable or Static Factors

Factor	Tunable	Consideration when Tuning	Restripe
Block size	No		
I-node size	No		
Indirect block size	No		
Number of I-nodes	No		
Max number nodes per file system	No		
Nodes available to file system	Yes	<Max number nodes per file system	Yes
Automount file system	Yes		
Max number metadata replicas	No		
Def number metadata replicas	Yes	=<Max number metadata replicas	Yes
Max number data replicas	No		
Def number data replicas	Yes	=<Max number data replicas	Yes
Stripe method	Yes		Yes
Number of disks	Yes		Yes
Number of nodes in GPFS pool	Yes	Quorum	Yes
GPFS starts on boot	Yes	Switch to be available	
Change to GPFS cache	Yes	Restart GPFS	
Quotas on/off	Yes		

2.2.1.1 Planning Your Nodes

You must consider the following when planning your nodes:

1. Nodelist

   You must prepare a list of the nodes you will run the GPFS daemon on. The collection of SP nodes is known as the GPFS pool. This list will not include any node that is a dedicated Virtual Shared Disk server. You do not have to run GPFS on a node that is acting as a VSD Node server. The work for GPFS is done at the Virtual Shared Disk device driver layer of the Virtual Shared Disk client.

   This list contains only one line per entry, and the list will contain the switch hostnames in any of the following formats: short hostname, long hostname, or IP address.

   Note

   Do not at this point identify the nodes by the Ethernet hostname, because this will result in degraded performance.

   An example of the creation of the nodelist is shown in 2.2.2.1, “Create Your List of SP Nodes Used in the GPFS System” on page 42.

   We use directory /var/mmfs as our directory for example configuration lists. If you do not specify a list of SP nodes when you configure GPFS, the switch hostnames of all the nodes in the System Data Repository (SDR) for the partition are added. These names are copied to the /etc/cluster.nodes file by GPFS. This file cannot be updated manually and must be identical on all your nodes.

2. Number of nodes available to a GPFS file system

   Each time you create a file system, you must supply the number of SP nodes available to a GPFS file system. When you create each file system, we recommand that you overestimate the number of nodes available to it. Include as many SP nodes as you are likely to expect in your GPFS file system, plus a margin for error. This information is used to optimize your file system; therefore, if you over-exaggerate this number, it will waste resources such as memory. The default value for this is 32 and this value cannot be changed dynamically.

3. Virtual Shared Disk to be a stand-alone server or not?

   All Virtual Shared Disks created will become available on each of the SP nodes known to the Virtual Shared Disks. The command `vsdatalst -n` shows these nodes. The optimum solution for a setup is to have a Virtual Shared Disk server on a different node from the SP nodes that have a GPFS file system mounted. The SP wide node at this time is the

best candidate to take on the responsibility of being a Virtual Shared Disk server. This will give the minimum amount of interference between disk processing and your applications.
Figure 6 shows a simple example setup utilizing Virtual Shared Disk, twin tailed disks, and the IBM Recoverable Shared Disk. This combination is also used as a configuration example in 2.2.2, “Steps of Configuration” on page 41:

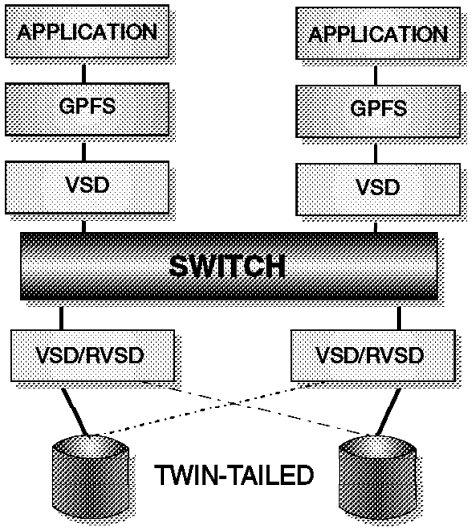


Figure 6. GPFS with Dedicated VSD Servers and RVSD Failover

4. Quorum

   Quorum for GPFS is performed on a SP node availability basis determined from the preceding nodelist. The quorum is based on an availability from “50 percent + 1” of your SP nodes. If quorum is not met, the GPFS system will not start. Also, if your file systems are mounted you will not be able to access your file systems until quorum is met again. You will have to consider this carefully when you are performing any maintenance work on your SP nodes. You must also take this into account when you add nodes. When a SP node is added to your GPFS system, it is seen as not operational and therefore can affect quorum.

   Following are examples of a successful and an unsuccessful addition of a SP node into the GPFS pool from our original four SP node example:

   - **Example 1**

The quorum for a four node SP system is three SP nodes available in the GPFS pool. We want to add two new nodes to our pool using mmaddnode. Once our SP nodes have been added, our new quorum would be four SP nodes. Therefore, because our original configuration had four SP nodes available to it, we would have a sufficient number of SP nodes available to continue functioning. This example would hold true for the addition of three nodes as well.

- **Example 2**

  Again our quorum for a four node SP system is three. This time we want to add four nodes into the pool. This attempt would fail because our new quorum would be set at five SP nodes. The original configuration only had four SP nodes available to the GPFS system.

  Therefore, to control this point of failure, the best solution is to add a maximum of three nodes at a time to your pool.

5. Starting the GPFS system automatically

   GPFS can be configured to start automatically when a SP node boots and then if the GPFS file system is set to mount from the daemon it will mount automatically as well. You can configure this when you initially configure GPFS. Automatic startup of GPFS is the recommended setting. This value is dynamic and can be changed at a later time.

6. Priority

   This is the priority of the daemon as set in the UNIX scheduler. The default value on the system is set to 40. We recommend that you do not change this value. The reason for this is that GPFS interacts with SP High Availability Services, and hatsd and hagsd have only marginally higher priorities as shown:

```
sp2n03> ps -el|grep mmfsd
     F S UID   PID  PPID   C PRI NI ADDR    SZ  WCHAN    TTY  TIME CMD
240503 A   0 19180  3524   0  40 -- 37ad  7324      *      -  0:34 mmfsd

sp2n03> ps -el|grep hagsd
340103 A   0 12914  3524   0  39 -- 1b06  3168            -  0:29 hagsd

sp2n03> ps -el|grep hatsd
4240103 A  0 11884  3524   0  38 -- 2709  9372      *      - 14:28 hatsd
```

7. Node Failure

   There are two issues to be considered in case of node failure: metadata integrity and userdata integrity. GPFS ensures metadata integrity by automatically protecting you against a non-Virtual Shared Disk server node failure. This is enabled by GPFS keeping two copies of its logs in

different failure groups. This log is the equivalent of a journal log in a JFS file system. Any data in memory will be lost, as it is in any file system.

The userdata integrity issue arises when your GPFS node is also a Virtual Shared Disk server node. The best scenario here is to use twin-tailed disks, which will enable you to utilize the facilities of the IBM Recoverable Shared Disk product to provide failover service to your disk, to a backup server. This is discussed more in depth in the following section.

2.2.1.2 Planning Your Disks and Failure Groups

1. Performance of Disks

   When planning your disks, you should plan to evenly distribute the disks across your Virtual Shared Disk server nodes. This will reduce the bottlenecks seen in I/O wait states. You should only have one Virtual Shared Disk per physical disk. This is the default option that is available to you when you let GPFS create your Virtual Shared Disks for you.

   You should also not have Virtual Shared Disks spanning several physical disks or several Virtual Shared Disks on one physical disk. This will only degrade performance and make your file systems more difficult to manage. The type of disk is important and it is generally recommended that you should be using any type of twin-tailed disk supported in AIX. The recommended setup would be to use SSA disks; this would be the ideal solution.

2. IBM Recoverable Shared Disk

   This product gives high availability to your volume groups and Virtual Shared Disks, and protects your disks against node failure. When a Virtual Shared Disk is created on a twin-tailed disk, either through GPFS or manually, you have to specify a primary and backup SP node.

   The Virtual Shared Disk is then created over a logical volume in a volume group. This volume group is imported on each of the Virtual Shared Disk nodes, but is only varied on and used on the primary node unless an SP node failure occurs.

   The failover process is a complicated process and follows a strict multiphase protocol to ensure data integrity. However, the basic activity is that the IBM Recoverable Shared Disk will make the primary node unavailable, and then vary on the volume group on the backup server and enable Virtual Shared Disk on that SP node.

3. Capacity

You should ensure that you have enough disk capacity to encompass your GPFS file system. The total size of disk space required, as with the AIX Logical Volume Manager, also depends on the degree of replication, the type of replication, AIX mirroring and so on that you are implementing. File system considerations are discussed in more detail in 2.2.1.4, “File System and File Size Considerations” on page 40.

4. Disk Failure

   You can protect yourself against disk failure as you would in a normal AIX environment: by using AIX mirroring or a RAID disk subsystem. For a highly available environment, you might consider using three mirrors.

   Replication is the tool available to you from GPFS to protect your data. There are two types of replication:

   - MetaData replication

     MetaData replication is used to replicate the *control pointers* of the system. MetaData consists mainly of i-node and indirect block information. The default for replication on the system is one, therefore no replication occurs. You can have a maximum replication of two.

   - Data replication

     Up to two copies of your data can be stored on your disks.

   For each additional copy of your data, you must use a different failure group; this will protect your data. Failure groups are described in more detail in 6 on page 38.

   However, there are considerations that you have to make regarding performance. Multiple copies have to be written and maintained. Replication by its very nature limits your maximum file size and inherently uses more disk space.

5. Striping

   There are three striping methods available to you in GPFS:

   The first method of striping is roundRobin, as shown in Figure 7 on page 36.

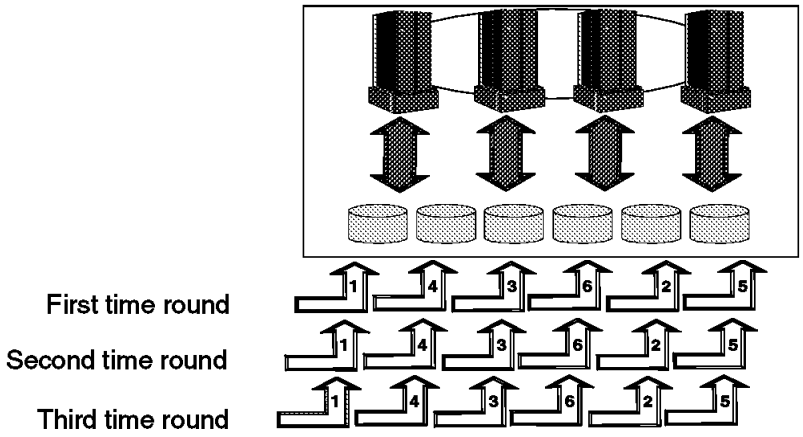


Figure 7. View of roundRobin Striping

This is the default option the system will select, and it is recommended because it gives the best performance. However, if you add or remove a disk, this option will take the longest of the three to restripe due to the uniform nature of the stripe. Data blocks are written to one disk at a time until all disks have received a block. The next round of writes will then again write a block to each disk and access the disks in the same order as the first round.

The second method is balancedRandom, as shown in Figure 8 on page 37.

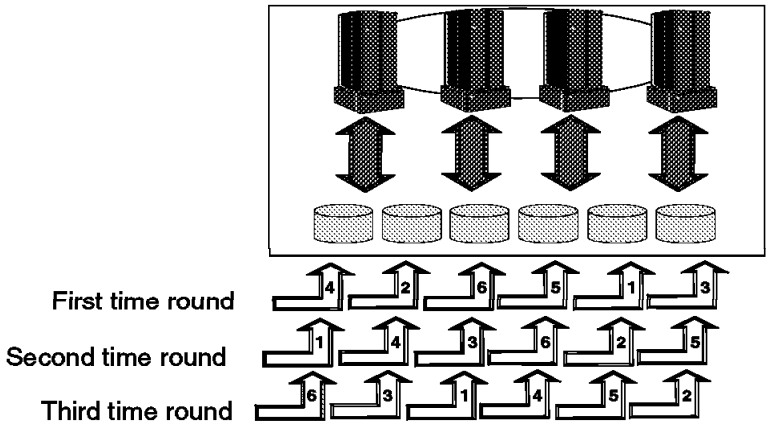


Figure 8. View of balancedRandom Striping

This option is similar to the first; however, this time the second (and any subsequent) pass will not write to the disks in the same order. It will not return to the same disk until all disks have been written to.

The last method available to you is random, as shown in Figure 9.

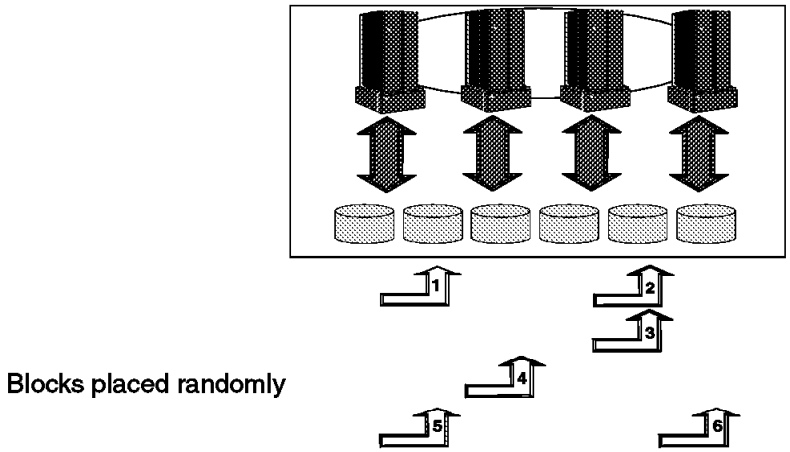


Figure 9. View of Random Striping

As the title suggests, this is a purely random function. This means that the GPFS system will write to any disk in any order and could possibly write to the same disk sequentially. If you are using replication and you have your failure groups set up, it will not write to the same disk twice for a replicated copy.

6. Failure Groups

   A *failure group* is a group of disks that have a common point of failure. GPFS by default will assign a failure group at the node level for each disk in the GPFS file system. The reason for this is that the node is seen as a single point of failure. You will be able to assign a number to each failure group.

   Another type of failure group that is seen from more than a single nodes point of view is a Virtual Shared Disk that is twin-tailed and available from two nodes over an IBM Recoverable Shared Disk. Although there are two nodes, this represents a single failure group.

   The default number that GPFS assigns to a failure group is the node number plus 4000. If you decide that a disk should ignore failure group

consideration, then you can assign it a value of -1. This generally indicates that a disk has no point of common failure with any other disk. You will be able to assign a value from -1 to 4000 for your failure groups. Refer to 2.2.2.4, “Creating Disk Descriptor Files” on page 46 for examples of failure groups.

2.2.1.3 Cache and Memory Considerations

During installation you will have to consider how much memory you are going to reserve for cache. This value must not exceed 80% of real memory. However, this is an unrealistic value for SP nodes with small amounts of memory, because this is *pinned* memory. It is reserved only for the GPFS system and therefore cannot be accessed by other applications. These values are tunable and can be changed at a later time. They are also tunable on an SP node basis as well. You will have to restart GPFS if these values are changed.

Following are the definitions of this cache:

1. PagePool

   The pagepool is used primarily for storage of user data and indirect blocks. Increasing the size of this page pool allows GPFS to cache more data for access by applications. The value of this pool can range from 4MB to 512MB per node. The system default at configuration time for this value is 20M. When you specify this value, you must specify it with the letter M, for example, to set your pagepool to a value of 40 megabytes, you must enter 40M.

2. Mallocsize

   This is the area that caches your metaData and control structures. Applications that reuse files heavily will benefit from an increase in this cache. The range for this cache is from 2MB to a maximum of 512MB, with a default of 4M.

As GPFS uses a large amount of pinned memory, you must carefully consider the amount of memory that your SP nodes have. The amount of pinned memory that the GPFS daemon utilizes is `mallocsize + pagepool`. If you take the system defaults, then `mallocsize=20M` and `pagepool=4M`, and then pinned memory for GPFS is 24MB.

Earlier, we had also set aside an amount of pinned memory for the switch. This was requested in the `rpoolsize` and `spoolsize`, with each being set to 16 megabytes. Therefore, with the GPFS defaults and switch cache, we already have 56MB set aside just for pinned memory.

More details about GPFS memory usage are discussed in *General Parallel File System for AIX: Installation and Administration,* SA22-7278. We recommend that you have at minimum 128MB of memory for a Virtual Shared Disk server node, but you should consider using 256MB and higher for your GPFS servers. However, use your knowledge of your own application to make the correct decision here.

2.2.1.4 File System and File Size Considerations

The GPFS file system is based on a standard UNIX file system architecture using i-nodes, indirect blocks, sub-blocks, and full data blocks. Refer to *General Parallel File System for AIX: Installation and Administration*, SA22-7278 and 1.2.3, “Allocation Mechanism” on page 7 in this redbook for detailed information about these features. The file system offers three block sizes available: 16KB, 64KB, and 256KB. This gives you a wide scope of choices for your data′s file size.

In this section we examine two scenarios:

1. Small Files

   If you have small files, you should consider using a 16KB block size. With this size, you waste as little disk space as possible when storing your data and you get the best performance for a file system with an average file size of 16KB and less. The data block is split into 32 sub-blocks, therefore giving you the ability to store more than one file in a full data block. These files can be as small as one sub-block in size; in the case of a 16KB, the sub-block would be 512 bytes in size. A system that stores small files should use the minimum i-node size of 512 bytes. This would keep the loss of disk space to a minimum.

2. Large Files

   When you require large amounts of data in a single read or write operation, you should opt for the larger block size and also larger indirect block sizes. This will give you the largest amount of contiguous disk space. Larger i-node sizes and indirect block sizes will result in wasted space if small files are written.

   If your file sizes are mixed, you should consider one of two options:

   - You could choose a block size of 64MB, which would balance the loss of disk space against reasonable performance from contiguous disk space.
   - You could create more than one file system to support the different file sizes. (This second option also relies heavily on disk availability.)

If you choose to replicate your system, logically you will see a decrease in the amount of space available to you in the file system. Refer to Appendix B, “GPFS Maximum File Size and Related Parameters” on page 189 to see what effects the different i-node sizes, block, indirect block sizes and replication has on your maximum file size and maximum file system size.

2.2.2 Steps of Configuration

We will now use the preceding considerations to build a GPFS system and make its file system available for use. We will be using four SP thin nodes and one SP wide node for this example configuration and each SP node will have 256mb of memory. Two SP nodes will be Virtual Shared Disk servers and the other three SP nodes will run the GPFS system and mount the file system. The five SP nodes sit in a partition named “sp2en1.” The SP nodes are named; sp2n03, sp2n04, sp2n07, sp2n08 and sp2n09 respectively. We have four SSA disks available to us; hdisk1, hdisk2 are 2.2 gigabyte drives and hdisk3, hdisk4 are 4.5 gigabyte drives. Note that hdisk3 and hdisk4 will be mirrored using AIX.

Note: All commands for GPFS administration must be run as root user and must be run on an SP node or from dsh.

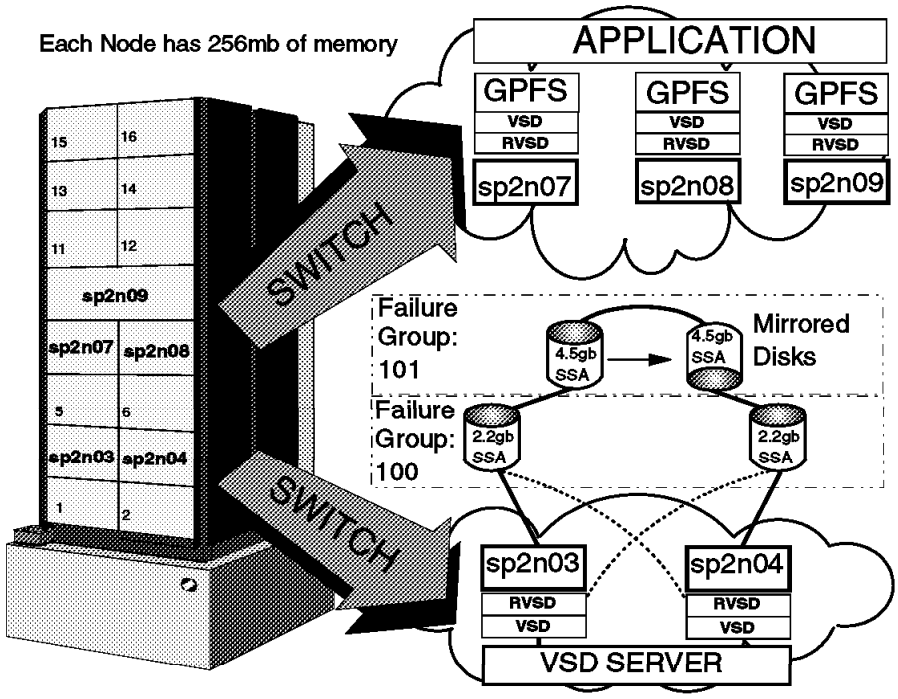


Figure 10. Example Configuration

Figure 10 shows the setup that will be used to define this GPFS system:. We will now see how this can be implemented on a GPFS system. All administration and configuration commands will be run on a SP node in your GPFS pool of nodes.

2.2.2.1 Create Your List of SP Nodes Used in the GPFS System

As discussed earlier, you must make available to the GPFS system a list of SP nodes that you will use. In our example, we create this file in directory /var/mmfs and call it nodes.pool. From our configuration figure, we can see that only three of our nodes will run the GPFS system; therefore, our list will only hold these three nodes. (You can, of course, add the other two nodes into the configuration later.)

We opt to use the short hostnames for our list. As discussed earlier, you can use a short hostname, long hostname or an IP address. The output from /var/mmfs/nodes.pool is as follows:

```
sp2n07> cat /var/mmfs/nodes.pool
sp2sw07
sp2sw08
sp2sw09
```

2.2.2.2 Configuring GPFS with Your Nodelist File

We now use the nodelist file to configure the GPFS system. We configure the system with default values, with one exception: we specify that the GPFS daemon should *autostart* on boot. If this is selected, then the GPFS daemon will not start until the IBM Recoverable Shared Disk and the switch are available. We can use one of two options here. We can use the `mmconfig` command from the command line:

```
sp2n07> mmconfig -n /var/mmfs/nodes.pool -A
```

Altertatively, we can use SMIT to configure GPFS. From the command line we can use a fast path to get to the GPFS main menu using:

- smitty gpfs

This will produce the following output:

```
                              GPFS Administration

Move cursor to desired item and press Enter.

  GPFS Configuration
  GPFS Change Configuration
  Prepare Disk Description List File
  GPFS Create File System
  GPFS Modify File System
  GPFS Display status of Disks
  GPFS Add Disk(s)
  GPFS Delete Disk(s)
  GPFS Replace Disk
  GPFS Add Node(s)
  GPFS Delete Node(s)
  GPFS Delete File system



F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

From there we can select **GPFS Configuration**. The fast path to get to this SMIT panel is:

- smitty mmconfig

You will be presented with the following screen:

```
                              Config GPFS File Sys

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
  node_file                                        [/var/mmfs/node.pool]     /
  config_file                                      []                        /
  Pagepool                                         []
  mallocsize                                       []
  priority                                         []                        #
  Auto Load                                    yes                 +
  client_ports                                     []                        #
  server_port_num                                  []                        #
  server_kprocs                                    []                        #



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The defaults of this setup are found in the sample configuration file:

- /usr/lpp/mmfs/samples/mmfs.cfg.sample

The details of the configuration have been entered into a System Data Repository (SDR) file. This SDR file is placed in directory /spdata/sys1/sdr/partitions/<ip_addr>/files and is called mmsdrcfg1. This file cannot be directly edited. You will have to use `SDRRetrieveFile mmsdrcfg1 <output_filename>` to view this file. The output from this file will be almost identical to the sample file /usr/lpp/mmfs/samples/mmfs.cfg.sample.

The SDR file is read every time the GPFS daemon is started on a SP node to define its working environment. There is a second SDR file of importance, and this is described in 2.2.2.3, "Starting Your GPFS Daemon."

2.2.2.3 Starting Your GPFS Daemon

Now that we have configured our GPFS nodes into the SDR, we are able to inform the GPFS daemon of its environment, and more importantly, the daemon will be able to work out its quorum. In our example we need to have two nodes out of the possible three available at all times for the GPFS quorum to be met. Before we start the GPFS system, however, the following two requirements must be meet:

1. The switch must be available.

   Use spmon or perspectives to check that the switch is available for the GPFS nodes; `spmon -d` is the quickest method to check this. You should also make sure that availability services are established by running:

   ```
   sp2n07> lssrc -l -s hags
   ```

   You should see two entries for *cssMembership* and *ha.vsd*. If you cannot start your switch, or if availability services are not available, then refer to *RS/6000 SP: PSSP 2.2 Survival Guide,* SG24-4928 for further information. If *ha.vsd* is not listed, then see the next heading.

2. The IBM Recoverable Shared Disk must be running.

   You also need to have your RVSD daemon running, and you can check this on each node by running:

   ```
   sp2n07> dsh lssrc -g rvsd
   sp2n07: Subsystem         Group            PID     Status
   sp2n07:  rvsd             rvsd             23670   active
   sp2n07:  hc.hc            rvsd             28692   active
   sp2n08: Subsystem         Group            PID     Status
   sp2n08:  rvsd             rvsd             27828   active
   sp2n08:  hc.hc            rvsd             31432   active
   ```

   At this point the working collective has been exported to the nodelist:

```
sp2n07> export WCOLL=/var/mmfs/nodes.pool
```

If this resource is unavailable to group services or the daemon is inactive, check the /etc/inittab for entry:

```
rvsd:2:once:/usr/lpp/csd/bin/ha_vsd > /dev/console 2>&1
```

If this entry exists and your rvsd daemon is inactive, you probably have not rebooted at this point. You can run the following command to perform a reboot:

```
sp2n07> dsh /usr/lpp/csd/bin/ha_vsd reset /dev/console 2>&1
```

Notes:

a. In our configuration, we need the IBM Recoverable Shared Disk to be available on the Virtual Shared Disk server because we use twin-tailed disks for the file system configuration.

b. You need to be careful running the `ha_vsd` command if you already have Virtual Shared Disks active on your system, since this will recycle your Virtual Shared Disks. This may be the case if you already use Virtual Shared Disk and not IBM Recoverable Virtual Shared Disk.

Both groups, cssMembership and ha.vsd, should now appear in the output from `lssrc -l -s hags`.

Now that we have the prerequisites for the GPFS system available, we can start the daemon:

```
sp2en7> dsh startsrc -s mmfs
```

To check that the daemon has started successfully on each node, you can check the mmfs log. The file takes two naming formats and sits in directory /var/adm/ras.

When the daemon is starting, the file is called mmfs.log. If this file continually exists, then you will experience problems starting up. The file size will also be seen to cycle from a zero length.

If the daemon is successful and has started, you will see a new log in the same directory similar to mmfs.log.1997.09.21.13.36.28. The log will look like the following:

```
Warning: Host has multiple network interfaces
Using interface 192.168.13.7
Use -h option to override
/usr/lpp/mmfs/bin/installcetm: extension is already loaded
MMFS: 6027-506 /usr/lpp/mmfs/bin/mmfskxload: /usr/lib/drivers/mmfs is already
loaded at 219566092.
MMFS: 6027-310 mmfsd initializing ...
MMFS: 6027-300 mmfsd ready for sessions.
```

The line `MMFS: 6027-300 mmfsd ready for sessions.` indicates that a quorum has been met and the GPFS system is now available.

2.2.2.4 Creating Disk Descriptor Files

Now that the GPFS system is available, we can create our file system. We call our file system `/dev/fs1` and we mount this file system over mount point `/gpfs/fs1`. Refer to Figure 10 on page 41 and 2.2.2, “Steps of Configuration” on page 41 to see see the disks we use in this file system.

There are four ways to add disks into a GPFS system:

1. Prepare a disk descriptor file through SMIT
2. Prepare a disk descriptor file through an editor like `vi`
3. Enter a list of disks into the SMIT **GPFS Create File System** option
4. Create a Virtual Shared Disk descriptor file

In the following sections, we describe these methods in more detail.

The first two methods create a file, which we call either /var/mmfs/disk.desc or /var/mmfs/vsd.desc. For our example, we create the file system *without* previsouly created Virtual Shared Disks. We use the first method to show the creation of a file, and the second method to explain the fields involved, as follows:

1. Prepare a disk descriptor file through SMIT.

   Enter `smitty gpfs` to get to the main menu, and then choose option **Prepare Disk Description List File**. Then select **Create** for the first pass and you will be asked to enter a file name that will be used in the creation of your file. We choose /var/mmfs/disk.desc. The following screen is then presented and we entered the following:

```
                        Disk Description List Dialog

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  #  Disk Description List File Name                 /var/mmfs/disk.desc
  #  Mode of operation on the DescFile               Create
  #  DiskDesc's Line no
* Hdisk Name                                        [hdisk1]                +
  Server Name                                       [sp2n03]                 +
  Backup Server Name                                [sp2n04]                 +
  Diskusage                                          dataAndMetadata         +
  Failure Group                                     [100]                    #




F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Once this data is entered, you must enter your second disk. This time we added hdisk2 into the same failure group. You will have to *Append* to the disk descriptor list this time; notice the change in the Mode field, as shown:

```
                        Disk Description List Dialog

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  #  Disk Description List File Name                 /var/mmfs/disk.desc
  #  Mode of operation on the DescFile               Append
  #  DiskDesc's Line no
* Hdisk Name                                        [hdisk2]                +
  Server Name                                       [sp2n04]                 +
  Backup Server Name                                [sp2n03]                 +
  Diskusage                                          dataAndMetadata         +
  Failure Group                                     [100]                    #




F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

We are now going to add our final disk known to GPFS into the descriptor file. As originally stated, this disk will be mirrored and sit in a separate failure group, which will enable us to use replication in our file system. There are two options in AIX mirroring. You can mirror at

the Logical Volume (LV) level, or you can now mirror at the Volume Group level itself, which makes administration an easier task in some scenarios. Again, you will need to *Append* to the disk descriptor file as follows:

```
                        Disk Description List Dialog

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
  #  Disk Description List File Name                /var/mmfs/disk.desc
  #  Mode of operation on the DescFile              Append
  #  DiskDesc's Line no
* Hdisk Name                                       [hdisk3]                +
  Server Name                                      [sp2n03]                 +
  Backup Server Name                               []                  +
  Diskusage                                         dataAndMetadata         +
  Failure Group                                    [101]                    #




F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Note: We did not use a backup server name because it is not available to the IBM Recoverable Shared Disk. We are also using the default disk usage *dataAndMetadata* to store the data. This is the recommended method of storing data.

You can, of course, split the metadata and data, but you will have to be extremely careful in setting up your failure groups and your high availability scenarios to keep the file system on-line. The most likely scenario for this is when you are using RAID. RAID performs at its optimum when you are using large block sizes. Therefore, with metadata being only a fraction of the block size, you may want to place your metadata on another type of disk (for example, a SSA disk). In this way, the RAID disks would only have data on them.

Nevertheless, we recommend that the SSA disk supporting the RAID disk with metadata should also hold a fraction of the data.

2. Prepare a disk descriptor file through an editor like vi.

   You can create this disk descriptor file manually. If we take a look at the file, we can see how the preceeding fields correspond:

```
hdisk1:sp2n03:sp2n04:dataAndMetadata:100
hdisk2:sp2n04:sp2n03:dataAndMetadata:100
hdisk3:sp2n03::dataAndMetadata:101
```

   Let us break down the definition of each field:

```
Disk Name:Server Name:Backup Server Name:Data Replication Type:Failure Group
hdisk1:   sp2n03:     sp2n04:            dataAndMetadata:      100
```

   a. Disk name

      This is the name of your device on which you wish to create the Virtual Shared Disk. As mentioned before, it is recommended that you have one physical disk per Virtual Shared Disk. If you have previously created your Virtual Shared Disks, you can also enter the name here.

      The creation of a descriptor file for Virtual Shared Disks is described in step 2d.

   b. Server name

      This the name of the Virtual Shared Disk server node.

   c. Backup server name

      This field is used when you have twin-tailed disks available and you require the use of IBM Recoverable Shared Disk to protect your access to the disk. It is the name of the secondary Virtual Shared Disk server node. The example shows that hdisk1 will be made available to sp2n04 using IBM Recoverable Shared Disk in the event of failure of sp2n03.

   d. Data replication type

      There are three types available to you:

      1) dataAndMetadata (This is the default and is the recommended data replication type.)

         This is the type we chose for our example configuration.

      2) dataOnly

      3) MetadataOnly

   e. Failure Group

We chose two failure groups because we require replication of hdisk1 and hdisk2 in failure group 100, to mirrored hdisk3 in failure group 101. If you have more than two failure groups, the replica will not necessarily be written to the same disk; in this case, GPFS will decide how to protect your data.

GPFS is able to work out the defaults. However, for a file that describes physical disks and not Virtual Shared Disks, you will always be required to enter the server name for the disk as shown:

```
hdisk3:sp2en3:::101
```

We also could have defaulted the failure group, since this disk is in a group of its own.

3. Enter a list of disks into the SMIT **GPFS Create File Sys** menu.

   When you select the **GPFS Create File System** option, you are offered two choices. Firstly, you can choose:

```
Get Disk Descriptions From                    DescFile
```

   which is the choice for the first two descriptor files; alternatively you can choose:

```
Get Disk Descriptions From                    DiskDescList
```

   and this will require you to add a string of your disks and definitions directly to the SMIT screen. This process is tricky as you are entering a lot of data. Our string looked like this:

   "hdisk1:sp2n03:sp2n04:dataAndMetadata:100**;**hdisk2:sp2n04:sp2n03:dataAndMetadata:100**;**hdisk3:sp2n03::dataAndMetadata:101**"**

   Your view from the SMIT screen is limited to only a set number of characters. As you can see, this is a not a simple operation and it is prone to errors if you are entering a sizable number of disks and descriptors.

4. Create a Virtual Shared Disk descriptor file.

   If you have already created your Virtual Shared Disks, you can add these disks to the GPFS file system by creating another descriptor file. You only need to enter three out of the five fields: disk name (nn in this case is the Virtual Shared Disk name); data replica type, and failure group.

Do not stipulate the server or backup server node, because the creation of the file system will fail. A Virtual Shared Disk descriptor file looks like the following:

```
gpfs1n3:::100
gpfs2n4:::100
gpfs3n3:::101
```

2.2.2.5 Creating a GPFS File System from a Disk Descriptor File

You can now use your disk descriptor file to create a file system. You can enter the GPFS Create File System menu by typing:

```
smitty mmcrfs
```

The following example uses the Physical Disk Descriptor File. You must select the following:

```
Get Disk Descriptions From                          DescFile
```

You are then presented with a new SMIT panel from which you are asked to enter details about your GPFS file system. We select the following values:

```
                              Create GPFS File Sys

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* mountpoint                                         [/gpfs/fs1]
* File System devicename                             [fs1]
* DescFile Name                                      [/var/mmfs/disk.desc]     /
  NumNodes                                           [16]                    #
  StripeMethod                                        roundRobin              +
  NumInodes                                          []                       #
  InodeSize                                          [512]                    #
  AutoMount                                           yes                     +
  BlockSize                                           256K                    +
  IndirectSize                                       [16384]                  #
  VerifyStripeGroup                                   yes                     +
  Activate Quotas                                     no                      +

F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

The highlighted entries denote the values we entered into the SMIT panel. In this GPFS version, we are unable to enter replication values through SMIT; therefore, this file system will have no replication after creation.

The successful creation of the file system looks like the following:

```
                                COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

[TOP]
mmcrfs: 6027-1052 Making MMFS filesystem /dev/fs1
createvsd -n 3/4:hdisk2/ -s 2148 -g gpfs2 -v gpfs -T 4 -o nocache
OK:0:defvsd lvgpfs1n3 gpfs2n3b4 gpfs1n3 nocache
mkvsdOK gpfs1n3 2148
createvsd -n 3/4:hdisk3/ -s 2148 -g gpfs3 -v gpfs -T 4 -o nocache
OK:0:defvsd lvgpfs2n4 gpfs3n4b3 gpfs2n4 nocache
mkvsdOK gpfs2n4 2148
createvsd -n 3/:hdisk4/ -s 4296 -g gpfs4 -v gpfs -T 8 -o nocache
OK:0:defvsd lvgpfs3n3 gpfs4n3 gpfs3n3 nocache
mkvsdOK gpfs3n3 4296
/usr/sbin/tscrfs /dev/fs1 -F /tmp/mmcrfsdddo30070 -M 1 -n 32 -R 1 -s roundRobin
-c 0 -p no -w 0

MMFS: 6027-531 The following disks of fs1 will be formatted:
    gpfs1n3: size 2199552KB
    gpfs2n4: size 2199552KB
    gpfs3n3: size 4399104KB
MMFS: 6027-540 Formatting file system ...
MMFS: 6027-572 Completed creation of file system /dev/fs1.

[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F6=Command
F8=Image            F9=Shell            F10=Exit            /=Find
n=Find Next
```

Following are the command and command line attributes that are facilitated with this SMIT:

```
mmcrfs /gpfs/fs1 fs1 -F /var/mmfs/disk.desc -A yes -n 16
```

We additionally have asked that the file system will automount on the startup of the GPFS system, and also that we have the maximum number of nodes available set to 16 since the GPFS system is unlikely to grow outside one frame.

The GPFS file system can also be created using a disk descriptor file that contains a list of Virtual Shared Disks, as already seen. The output from SMIT on a successful creation of the file system is a little different and looks like the following:

```
                                COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

mmcrfs: 6027-1052 Making MMFS filesystem /dev/fs1
/usr/sbin/tscrfs /dev/fs1 -F /tmp/mmcrfsdddo31996 -M 1 -n 32 -R 1 -s roundRobin
-v no -c 0 -p no -w 0

MMFS: 6027-531 The following disks of fs1 will be formatted:
    gpfs1n3: size 2199552KB
    gpfs2n4: size 2199552KB
    gpfs3n3: size 4399104KB
MMFS: 6027-540 Formatting file system ...
MMFS: 6027-572 Completed creation of file system /dev/fs1.



F1=Help             F2=Refresh          F3=Cancel           F6=Command
F8=Image            F9=Shell            F10=Exit            /=Find
n=Find Next
```

Except for the file name change, the command line arguments for the mmcrfs command are identical to the previous example.

The example configuration of a GPFS file system requires replication. As previously mentioned, we cannot add these values from the SMIT panel; therefore, we have to use the command line option.

Note: You cannot change MaxDataReplicas and MaxMetaDataReplicas *after* file system creation. These values are used in the calculation of your maximum file size and maximum file system size.

We require two replicas of data and metadata. Therefore, the additional arguments for the mmcrfs command are:

```
-M 2 -m 2 -R 2 -r 2
```

The uppercase arguments in this case denote the MAX value, and the lowercase arguments denote the actual replication at the time of file system creation. The complete command would therefore look as follows:

```
mmcrfs /gpfs/fs1 fs1 -F /var/mmfs/disk.desc -A yes -n 16 -M 2 -m 2 -R 2 -r 2
```

The output from the command is similar to that seen in the SMIT version.

2.2.2.6 Mounting and Checking the GPFS File System

The mounting of a GPFS file system is the same as mounting any file system: you simply use the mount command on each GPFS node that you have in the GPFS pool, as shown:

```
sp2n07> dsh mount /gpfs/fs1
```

The GPFS system manages the stanza that is seen for the file system in /etc/filesystems:

```
/gpfs/fs1:
        dev             = /dev/fs1
        vfs             = mmfs
        nodename        = -
        mount           = mmfs
        type            = mmfs
        options         = rw,disks=gpfs1n3;gpfs3n3;gpfs2n4
        account         = false
```

Do not modify the stanza manually, because it is changed by the GPFS daemon and will be overlaid the next time the SDR copy is fetched.

You can now check that the file system is mounted by entering the traditional df command, which will show you the following:

```
sp2n07> df
Filesystem    512-blocks      Free %Used    Iused %Iused Mounted on
/dev/hd4           16384      2848   83%     1238    31% /
/dev/hd2          532480      5120  100%     8920    14% /usr
/dev/hd9var       106496     44104   59%      507     4% /var
/dev/hd3          131072     46240   65%       90     1% /tmp
/dev/hd1            8192      7840    5%       18     2% /home
/dev/fs1        17596416  17559040    1%       10     1% /gpfs/fs1
```

Alternatively, you can run the GPFS equivalent mmdf:

```
sp2n07> mmdf fs1
disk            disk size  failure holds    holds    in full        in
name                in KB    group metadata data      blocks fragments
--------------- --------- -------- -------- ------ --------- ---------
gpfs1n3           2199552      100 yes      yes      2194432       416
gpfs2n4           2199552      100 yes      yes      2194688       416
gpfs3n3           4399104      101 yes      yes      4389376       648
                 ---------                          --------- ---------
(total)           8798208                         8778496      1480
```

The following screen shows how GPFS views the disk and availability information:

```
sp2n07> mmlsdisk fs1
disk         driver   sector failure holds    holds
name         type       size   group metadata data  status        availability
------------ -------- ------ ------- -------- ----- ------------- ------------
gpfs1n3      disk        512     100 yes      yes   ready         up
gpfs2n4      disk        512     100 yes      yes   ready         up
gpfs3n3      disk        512     101 yes      yes   ready         up
```

You can also check your Virtual Shared Disks at this point:

```
sp2n07> lsvsd -l
minor  state server lv_major lv_minor vsd-name             option   sizee(MB)
11      ACT    3       0       0      gpfs1n3              nocache  21488
12      ACT    4       0       0      gpfs2n4              nocache  21488
17      ACT    3       0       0      gpfs3n3              nocache  42966
```

In addition, you can also use the GPFS command, `mmlsfs`, to list your file system attributes:

```
sp2n07> mmlsfs fs1
flag value          description
---- -------------- -----------------------------------------------------
 -s  roundRobin     Stripe method
 -f  8192           Minimum fragment size in bytes
 -i  512            Inode size in bytes
 -I  16384          Indirect block size in bytes
 -m  2              Default number of metadata replicas
 -M  2              Maximum number of metadata replicas
 -r  2              Default number of data replicas
 -R  2              Maximum number of data replicas
 -a  1048576        Estimated average file size
 -n  16             Estimated number of nodes that will mount file system
 -c  0              Maximum concurrent I/O operations per disk
 -d  gpfs1n3;gpfs2n4;gpfs3n3  Disks in file system
 -B  262144         Block size
 -Q  no             Quotas on|off?
```

The file system has now been created and mounted and is available for use. GPFS has created and configured your Virtual Shared Disks for you and made them available to the IBM Recoverable Virtual Shared Disk. You will now have to use AIX mirroring to complete the configuration for hdisk3 and hdisk4. Other example configurations are available in *General Parallel File System for AIX: Installation and Administration*, SA22-7278.

Chapter 3. Failover Scenarios

In order to ensure a highly available file system, GPFS provides capabilities that enable it to cope with hardware/software failures that can occur in the system.

This chapter discusses some of the failover scenarios that may happen with GPFS and describes how GPFS reacts to these failures.

3.1 Hardware Recovery

In this section, we describe the hardware failures that can happen and how GPFS handles the recovery in each case.

The three hardware components we discuss are:

- The node itself
- The switch connection between nodes
- The disk subsystem

3.1.1 Node Failure

A node in a GPFS file system domain can be classified into:

1. VSD server node
   - VSD server primary node
   - VSD server secondary node
2. Client node

A *Virtual Shared Disk* (VSD) is a logical volume that can be accessed not only from the node it belongs to, but also from any other node in the system partition.

A VSD *server* is a node that owns a number of VSDs. It reads and/or writes data to VSDs as requested by client nodes, and transfers the data back, usually via SP Switch or HiPS Switch.

A *client* node is a node that requests access to VSDs. It should be noted that a node can be both a VSD server node and a client node at the same time.

If a VSD server node fails, access to the data on all VSDs that it owns is lost. In order to avoid this situation, we implement IBM Recoverable Virtual Shared Disk (RVSD) and twin-tailed or loop cabling between nodes.

The RVSD concept is to allow not only one node (the VSD server primary node) to have access to a set of VSDs, but also a second node (the VSD server secondary node), in case of one of the following fails:

- VSD server primary node
- Switch adapter
- Disk adapter
- Disk or network cables

We achieve this by twin-tailed or loop cabling the disk subsystems between the VSD server primary node and VSD server secondary node, as shown in Figure 11.

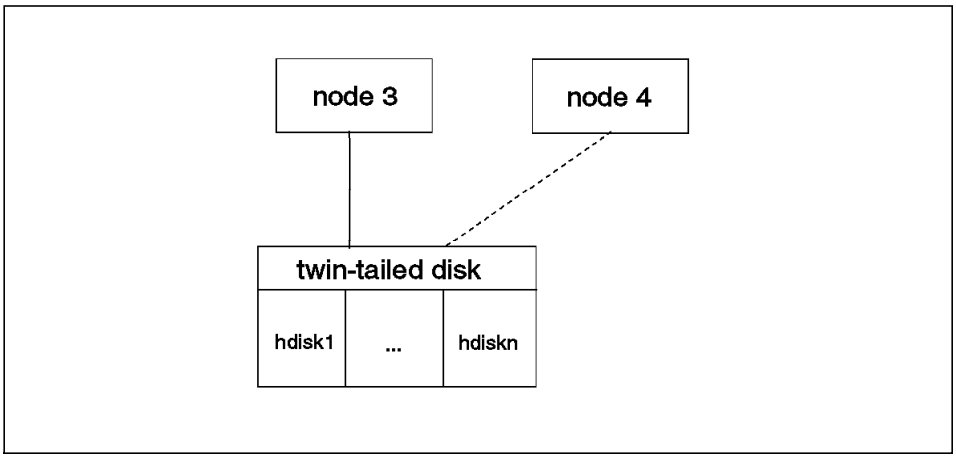


Figure 11. Twin-Tailed Configuration

Normally, the VSD server secondary node does nothing with those VSDs. In fact, it cannot access them since they are being held in use by the VSD server primary node.

RVSD provides protection against node failure by subscribing to Group Services. When a node fails, RVSD will be informed by Group Services.

If the failed node is a VSD server primary node, RVSD will have the VSD server secondary node for each VSD perform the necessary functions to ensure all VSDs on the failed node can still be accessed by the clients.

This is achieved by having the VSD server secondary node take over the disk subsystems and become the server for those VSDs while the primary node is unavailable.

When the VSD server primary node comes back, you must use the vsdchgserver command to switch the server function back to the primary node yourself.

Thus, with RVSD, the disk subsystem becomes highly available since you can have continuous access to the VSDs even when the VSD server primary node is down.

The amount of time required to take over depends mainly on how many disks RVSD needs to import onto the VSD server secondary node.

Note: RVSD is a prerequisite for GPFS.

You need RVSD even when you do not care about the high availability of your file systems, or do not plan to use any external disk subsystem. This is because GPFS needs some commands in RVSD, for example, fencevsd and unfencevsd, that are necessary to ensure that the integrity of the GPFS file system will not be compromised.

Table 3 shows some new RVSD commands that are available in PSSP.

Table 3. New RVSD Commands in PSSP

Command	Description	Usage example
fencevsd	To prevent a node from accessing VSDs.	GPFS uses this to prevent a GPFS-perceived failed node from accessing VSDs.
unfencevsd	To allow a node to access VSDs.	GPFS uses this to allow a previously failed node, that has completed the recovery process, to access VSDs.
vsdchgserver	To swap the duty of the primary node and secondary node for a global volume group.	When the disk adapter fails, RVSD switches the server function from the primary node to the secondary node automatically.
lsfencevsd	To show which VSDs are fenced from which nodes in a system partition.	After a node comes back from recovery, GPFS uses this to find out which VSDs has been fenced and will unfence them so that the node can have access to those VSDs as usual.

When you plan for GPFS configuration, it is highly recommended that you implement RVSD and twin-tailed or loop cabling between your VSD servers, so that the recovery of VSD servers will be done automatically by RVSD.

If you do not implement RVSD and also do not have replicas, some disks in the stripe group will not be available when a node is down. In this case whether or not you can still access the data in the file system depends on what data the missing disks contain.

Sometimes you may not be able to use the file system at all since some important metadata may reside on the missing disks.

For example, on the following screen we show four VSDs defined for the koafs file system on four disks, one on each node. When node 13 fails, the availability status of gpfsvsd3n13 becomes “down.”

Unfortunately, it seems that some important metadata is in this VSD, and hence we cannot use the koafs file system until node 13 comes up again.

Note: Whenever GPFS determines that continuing the operation may compromise the integrity of the file system, it will force unmount the file system so that no one can use it.

Another complication here is that, even when node 13 comes up and you have started GPFS, the koafs file system still cannot be used. You have to issue the mmchdisk command to start the disk before it can be used.

```
sp21n11:/ } mmlsdisk koafs
disk         driver   sector failure holds    holds
name         type       size   group metadata data  status        availability
------------ -------- ------ ------- -------- ----- ------------- ------------
gpfsvsd1n1   disk        512      -1 yes      yes   ready         up
gpfsvsd2n11  disk        512      -1 yes      yes   ready         up
gpfsvsd3n13  disk        512      -1 yes      yes   ready         down
gpfsvsd4n15  disk        512      -1 yes      yes   ready         up
sp21n11:/ }
sp21n11:/ } mmchdisk koafs start -d 'gpfsvsd3n13'
MMFS: 6027-560 Scanning file system metadata ...
MMFS: 6027-565 Scanning user file metadata ...
MMFS: 6027-552 Scan completed successfully.
sp21n11:/ }
sp21n11:/ } mmlsdisk koafs
disk         driver   sector failure holds    holds
name         type       size   group metadata data  status        availability
------------ -------- ------ ------- -------- ----- ------------- ------------
gpfsvsd1n1   disk        512      -1 yes      yes   ready         up
gpfsvsd2n11  disk        512      -1 yes      yes   ready         up
gpfsvsd3n13  disk        512      -1 yes      yes   ready         up
gpfsvsd4n15  disk        512      -1 yes      yes   ready         up
```

Notes:

1. A disk remains stopped until it is explicitly started by the mmchdisk command. Restarting the GPFS daemon or rebooting does not restore normal access to a stopped disk.
2. It is recommended that if you have to start more than one disk, specify them all in the same mmchdisk command.

In RVSD, only the secondary node performs the recovery for the primary node. This means that if the secondary node is down, RVSD takes no action and therefore, if the primary node fails later, there will be no node to take over.

However, we can use the Problem Management subsystem to prevent this from happening by creating a VSD server secondary node “node down” event that triggers a notification, so that an appropriate action can be taken to bring the secondary node back as soon as possible.

The following screen shows how to set up the Problem Management subsystem to detect when a VSD server secondary node fails.

```
sp2en1:/ ) vsdatalst -g
      VSD Global Volume Group Information
                                                Server Node Numbers
Global Volume Group name        Local VG name   primary      backup eio_recovery
     recovery
------------------------------- --------------- -------      ------ ------------
     --------
gpfsn3b4                        gpfs                  3         4
 0          0
rootvgn3                        rootvg                3         0
 0          0
sp2en1:/ )
```

We have a VSD (gpfsn3b4) that has node 4 as a backup server. We set up the Problem Management subsystem to notify all users on the Control Workstation whenever the rvsdd daemon on node 4 fails, as follows:

1. Set up for Sysctl.

   Since the Problem Management subsystem makes use of the Sysctl facility, you need to modify the file /etc/sysctl.pman.acl to allow root.admin to execute Problem Management subsystem-related commands.

   You also need to propagate this to every node in the system partition, as shown in the following screen.

```
sp2en1:/ ) cat /etc/sysctl.pman.acl
#acl#

# These are the kerberos principals for the users that can configure
# Problem Management on this node.  They must be of the form as indicated
# in the commented out records below.  The pound sign (#) is the comment
# character, and the underscore (_) is part of the "_PRINCIPAL" keyword,
# so do not delete the underscore.

#_PRINCIPAL root.admin@PPD.POK.IBM.COM
#_PRINCIPAL joeuser@PPD.POK.IBM.COM
_PRINCIPAL root.admin@MSC.ITSO.IBM.COM
sp2en1:/ )
sp2en1:/ ) pcp -a -p /etc/sysctl.pman.acl /etc/sysctl.pman.acl
sp2en1:/ )
```

2. Define and subscribe the event to the Problem Management subsystem.

   Here we register a monitor for the rvsdd process on node 4. So, whenever this process on node 4 is down, the Problem Management subsystem will execute the wall command to send the message “rvsd daemon on secondary node 4 is down...” to all users on the Control Workstation.

```
sp2en1:/ ) pmandef -s RvsdNode4Down \
> -e 'IBM.PSSP.Prog.xpcount:NodeNum=4;ProgName=rvsdd;UserName=root:X@0==0' \
> -c 'wall rvsd daemon on secondary node 4 is down...' \
> -n 0 -U root
sp2en1:/ )
```

3. Verify the subscription.

```
sp2en1:/ ) lssrc -ls pman.sp2en1
Subsystem         Group            PID     Status
 pman.sp2en1      pman             60792   active

 pmand started at: Thu Sep 18 14:14:00 1997
 pmand last refreshed at:
 Tracing is off
 ========================================================
 Events for which registrations are as yet unacknowledged:
 ========================================================
 ========================================================
 Events for which actions are currently being taken:
 ========================================================
 ========================================================
 Events currently ready to be acted on by this daemon:
 ========================================================
 ------------------ RvsdNode4Down ----------------
 Currently ACTIVE
 Client root root.admin@MSC.ITSO.IBM.COM at sp2en0.msc.itso.ibm.c
 om
 Resource Variable: IBM.PSSP.Prog.xpcount
 Instance: NodeNum=4;ProgName=rvsdd;UserName=root
 Predicate: X@0==0
 Command to run: wall rvsd daemon on secondary node 4 is down...
 Has run 0 times
sp2en1:/ )
```

4. Test by simulating the event.

```
sp2en1:/ ) dsh -w t04 stopsrc -g rvsd
t04: 0513-044 The stop of the rvsd Subsystem was completed successfully.
t04: 0513-044 The stop of the hc.hc Subsystem was completed successfully.
sp2en1:/ )

Broadcast message from root@sp2en0 (tty) at 16:04:27 ...

rvsd daemon on secondary node 4 is down...


sp2en1:/ )
```

A failure of the client node can be addressed similarly.

Note

In case some special components of GPFS have been running on the failed node (for example, the Configuration Manager, the Stripe Group Manager), GPFS recovers these components automatically, so that the operation of GPFS can continue without any effect from the failure. (These recoveries are described in 3.2, “Software Recovery” on page 66.)

3.1.2 Network Failure

Currently, GPFS is supported only in SP frames, and only with an SP Switch or HiPS Switch as the underlying network connection between nodes.

When GPFS starts up, it checks whether the switch network is available. If the switch network is not available, it waits until it becomes available, and logs the following message in the file mmfs.log in the directory /var/adm/ras.

```
      6027-1242 mmfs is waiting for switch
      6027-1242 mmfs is waiting for switch
      6027-1242 mmfs is waiting for switch
      .....
```

As soon as the switch network becomes available, the startup process continues normally.

```
      6027-1242 mmfs is waiting for switch
      6027-1242 mmfs is waiting for switch
      6027-1242 mmfs is waiting for switch
Warning: Host has multiple network interfaces
Using interface 192.168.14.5
Use -h option to override
/usr/lpp/mmfs/bin/installcetm: /usr/lpp/mmfs/bin/cetm.kext loaded at 0x01aeeb88
Warning: Host has multiple network interfaces
Using interface 192.168.14.5
Use -h option to override
/usr/lpp/mmfs/bin/installcetm: /usr/lpp/mmfs/bin/sharkTM.kext loaded at 0x01b14c74
MMFS: 6027-310 mmfsd initializing ...
MMFS: 6027-300 mmfsd ready for sessions.
mounting /chico
mounting /gpfs/koa2
```

GPFS not only checks the availability of the switch network, but also the availability of RVSD and Group Services.

If RVSD is not available, it logs the following message:

```
      6027-1242 mmfs is waiting for rvsd
      6027-1242 mmfs is waiting for rvsd
      6027-1242 mmfs is waiting for rvsd
      .....
```

If Group Services is not available, it logs the following message:

```
6027-1242 mmfs is waiting for hags
6027-1242 mmfs is waiting for hags
6027-1242 mmfs is waiting for hags
.....
```

GPFS checks for these three services every six seconds. In order not to clutter up the log file, GPFS logs the message the first time it finds one of these not available. It then logs a message once a minute for five minutes. After that, it logs a message once every five minutes.

3.1.3 Disk Failure

Though RVSD can provide for higher availability in case of node failure and disk adapter failure, it does not provide any protection in case the disk itself fails.

There are several options we can implement to provide higher availability for the disk subsystem:

- Use of a RAID disk
- Mirroring
- GPFS replication

Using a RAID-5 disk subsystem can provide protection against a single disk drive failure with a minimal amount of additional investment. The limitation of this option is that its performance may not meet the requirement for applications that require a high level of performance.

Also, an additional disk adapter may be needed to provide protection against disk adapter failure. Moreover, twin-tailed or loop cabling may also be needed to provide additional protection against node failure.

Mirroring is an attractive option for applications that require high performance.

Similarly, an additional disk adapter may be needed to provide protection against disk adapter failure. Also, twin-tailed or loop cabling may be needed to provide additional protection against node failure.

Notes:

1. Since PSSP 2.3, RVSD can provide protection against disk adapter failure by automatically switching the server function to the VSD server secondary node when it detects that the disk adapter in the VSD server primary node has failed.
2. After replacing the failed adapter and powering on the node, you must use the vsdchgserver command to switch the server function back to the primary node yourself.

GPFS replication may be a good option for those who are not satisfied with RAID-5 disk performance and mirroring cost. You can get good performance from replication by using SSA disks, and you can minimize cost by replicating only important files.

You can select what you want to replicate: either the data or the metadata of a file system, or both. You can select up to two replicas for data and metadata.

Moreover, since GPFS automatically places each replica in different failure groups, this enables replication to tolerate the failure of a single disk adapter or a single node without any additional hardware. In certain cases, it may be even able to tolerate multiple disk adapter or node failures.

The limitation of replication is that when it is enabled, the maximum file system size and the maximum file size that can be created in the file system are reduced significantly. With a replica factor of 2, the maximum file system size and the maximum file size are reduced by an approximate factor of 4. Refer to Appendix B, “GPFS Maximum File Size and Related Parameters” on page 189 for more information.

Also, if the file system is heavily updated, the overhead due to updates to the replica may become significant and, in some cases, compromise the previously listed benefits.

3.2 Software Recovery

When a hardware failure occurs that causes a node to be down, mere hardware recovery is not enough to get the system back since some GPFS components that were running on that node became unavailable when the failure happened.

Therefore, in this case we also need to have software recovery. GPFS automatically handles this recovery to make sure that it still functions with no interruption to the end users even if failures occur.

In this section, we discuss the following components of GPFS:

- Configuration Manager
- Stripe Group Manager
- Metadata Manager
- Token Manager Server
- Token Manager

We describe what they are, what their responsibilities are, and how GPFS reacts and recovers in case they fail.

3.2.1 Configuration Manager Failure

Configuration Manager is a component of GPFS that is responsible for selecting the Stripe Group Manager for each file system. It also determines whether a quorum exists, which in turn determines whether the file system can continue to be used.

When a quorum is lost, GPFS unmounts the file system (thus not allowing it to be used), since there can be cases where problems with the network cause the GPFS domain to be divided into separate groups.

If GPFS does not force the group with no quorum to stop using the file system, many groups may try to write to the same metadata and/or file at the same time, compromising the integrity of the file system.

There is one Configuration Manager per system partition. It is the first node to join the group MmfsGroup in Group Services. In other words, it is the oldest node in this group.

To identify the Configuration Manager, issue the following command on the Control Workstation:

```
dsh sysctl mmremote clmgr
```

```
sp21en0:/ } dsh sysctl mmremote clmgr
n01: sp21n11.msc.itso.ibm.com
n05: sp21n11.msc.itso.ibm.com
n06: sp21n11.msc.itso.ibm.com
n07: sp21n11.msc.itso.ibm.com
n08: sp21n11.msc.itso.ibm.com
n09: sp21n11.msc.itso.ibm.com
n11: sp21n11.msc.itso.ibm.com
n13: sp21n11.msc.itso.ibm.com
n15: sp21n11.msc.itso.ibm.com
```

If the Configuration Manager is down, Group Services will select the next oldest node in MmfsGroup to become the Configuration Manager.

```
sp21en0:/ } dsh -w n11 hagsgr -s hags -a MmfsGroup
n11:   Number of: groups: 7
n11: Group slot # [5] Group name[MmfsGroup] group state[Inserted |Idle |]
n11: Providers[[1/11][1/9][1/13][1/5][1/8][1/6] ... ]
n11: Local subscribers[]
n11:
sp21en0:/ }
sp21en0:/ } dsh -w n11 stopsrc -c -s mmfs
n11: 0513-044 The stop of the mmfs Subsystem was completed successfully.
sp21en0:/ }
sp21en0:/ } dsh sysctl mmremote clmgr
n01: sp21n09.msc.itso.ibm.com
n05: sp21n09.msc.itso.ibm.com
n06: sp21n09.msc.itso.ibm.com
n07: sp21n09.msc.itso.ibm.com
n08: sp21n09.msc.itso.ibm.com
n09: sp21n09.msc.itso.ibm.com
n11: /usr/lpp/mmfs/bin/mmfsadm: mmfs daemon not running
n13: sp21n09.msc.itso.ibm.com
n15: sp21n09.msc.itso.ibm.com
sp21en0:/ }
sp21en0:/ } dsh -w n09 stopsrc -c -s mmfs
n09: 0513-044 The stop of the mmfs Subsystem was completed successfully.
sp21en0:/ }
sp21en0:/ } dsh sysctl mmremote clmgr
n01: sp21n13.msc.itso.ibm.com
n05: sp21n13.msc.itso.ibm.com
n06: sp21n13.msc.itso.ibm.com
n07: sp21n13.msc.itso.ibm.com
n08: sp21n13.msc.itso.ibm.com
n09: /usr/lpp/mmfs/bin/mmfsadm: mmfs daemon not running
n11: /usr/lpp/mmfs/bin/mmfsadm: mmfs daemon not running
n13: sp21n13.msc.itso.ibm.com
n15: sp21n13.msc.itso.ibm.com
sp21en0:/ }
```

By using the hagsgr command, you can get a list of all members in MmfsGroup. In this case, we see that node 11 is the first one to join the group, node 9 is the second, node 13 is the third and so on.

If we stop GPFS on node 11 (which is currently the Configuration Manager) by issuing the stopsrc command, then node 9 (which is the next oldest node in the group), will become the Configuration Manager.

Similarly, if we stop GPFS on node 9, we will see that node 13, the next oldest node in the group, becomes the Configuration Manager.

3.2.2 Stripe Group Manager Failure

Each GPFS file system is comprised of a *stripe* group. A stripe group is simply a set of disks that belong to this file system.

There is one Stripe Group Manager per file system. The Configuration Manager selects a Stripe Group Manager for each file system. It tries not to overload any node in the system by selecting a different node to act as a Stripe Group Manager for each file system, if possible.

The Stripe Group Manager provides the following services to all nodes using that file system:

- Processes changes to the state or description of the file system
  - Adds/deletes/replaces disks
  - Changes disk availability
  - Repairs the file system
  - Restripes the file system
- Controls disk region allocation

To identify the Stripe Group Manager, issue the following command on the Control Workstation:

```
dsh sysctl mmremote sgmgr <device>
```

```
sp21en0:/ } dsh sysctl mmremote sgmgr chico
n01: child process exited abnormally
n05: child process exited abnormally
n06: child process exited abnormally
n07: child process exited abnormally
n08: child process exited abnormally
n09: child process exited abnormally
n11: sp21n06.msc.itso.ibm.com
n13: child process exited abnormally
sp21en0:/ }
sp21en0:/ } dsh sysctl mmremote sgmgr koafs2
n01: child process exited abnormally
n05: child process exited abnormally
n06: child process exited abnormally
n07: child process exited abnormally
n08: child process exited abnormally
n09: child process exited abnormally
n11: sp21n07.msc.itso.ibm.com
n13: child process exited abnormally
sp21en0:/ }
```

Only the Configuration Manager is able to reply to this query. Nodes that are not the Configuration Manager respond with “child process exited abnormally.”

In this case, the Stripe Group Manager for file system chico is sp21n06 and the Stripe Group Manager for file system koafs is sp21n07.

If needed, you can influence the Configuration Manager regarding the selection of the Stripe Group Manager by creating a file called cluster.preferences in the /var/mmfs/etc directory and listing the nodes that you want to act as a Stripe Group Manager, as shown in the following example:

```
sp21en0:/ } dsh -w n11 cat /var/mmfs/etc/cluster.preferences
n11: sp21sw07
n11: sp21sw06
n11: sp21sw05
n11: sp21sw08
sp21en0:/ }
```

Note

Whenever possible, the Stripe Group Manager should not be running in a VSD server node, due to AIX kernel heap contention.

Notes:

1. The preference file tells the Configuration Manager to select the Stripe Group Manager from this list, if any of the listed nodes are available when the choice is made. There is no relative priority or rank among the nodes.
2. The hostnames listed in this file must be the *switch hostname* only. Using other hostnames, for example, the Ethernet one, will not work.

When a Stripe Group Manager is down:

- If you use the preference file, the Configuration Manager selects a node from that file to be the Stripe Group Manager. If possible, it selects a node that is not currently a Stripe Group Manager for any other file system.
- If you do not use the preference file, the Configuration Manager selects any node. If possible, it selects a node that is not currently a Stripe Group Manager for any other file system.

```
sp21en0:/ } dsh sysctl mmremote sgmgr koafs2
n01: child process exited abnormally
n05: child process exited abnormally
n06: child process exited abnormally
n07: child process exited abnormally
n08: child process exited abnormally
n09: child process exited abnormally
n11: sp21n07.msc.itso.ibm.com
n13: child process exited abnormally
sp21en0:/ }
sp21en0:/ } dsh -w n07 stopsrc -c -s mmfs
n07: 0513-044 The stop of the mmfs Subsystem was completed successfully.
sp21en0:/ }
sp21en0:/ } dsh sysctl mmremote sgmgr koafs2
n01: child process exited abnormally
n05: child process exited abnormally
n06: child process exited abnormally
n07: /usr/lpp/mmfs/bin/mmfsadm: mmfs daemon not running
n08: child process exited abnormally
n09: child process exited abnormally
n11: sp21n08.msc.itso.ibm.com
n13: child process exited abnormally
sp21en0:/ }
```

By using the `dsh sysctl mmremote sgmgr koafs2` command, you can identify which node is the Stripe Group Manager for file system koafs2.

You then use the `stopsrc -c -s mmfs` command to stop GPFS on node 7 (which is currently the Stripe Group Manager for koafs2) to simulate the Stripe Group Manager failure.

Later, when you use the dsh sysctl mmremote sgmgr koafs2 command again, you will find that the Stripe Group Manager for file system koafs2 has been changed from node 7 to node 8.

3.2.3 Metadata Manager Failure

There is one Metadata Manager for each open file in the file system. It is responsible for the integrity of the metadata of that file.

Even though each VSD server node can read and/or write the data to the disks directly, the update of metadata of a file is restricted to the node containing the Metadata Manager for that file.

To identify the Metadata Manager for a file, issue the following commands on the Control Workstation:

```
dsh -w <node> ls -il <file>
dsh -w <node> mmfsadm dump files|pg
```

```
sp21en0:/ } dsh -w n11 ls -il /chico
n11: total 43
n11:  1275 -rw-r--r--   1 chico    notes       1314 Sep 06 16:02 .Xpdefau
n11:   104 -rwxr--r--   1 chico    notes        331 Sep 06 15:54 .profile
n11:   348 -rw-------   1 chico    notes        214 Sep 08 11:52 .sh_hist
n11:    92 drwxr-xr-x   7 chico    notes       3072 Sep 06 16:38 notesr4
n11:   222 drwxr-xr-x   2 theeraph staff        512 Sep 08 12:44 theeraph
sp21en0:/ }
sp21en0:/ } dsh -w n11 mmfsadm dump files|pg
n11:
n11: Files in stripe group chico:
n11: ....
n11: ....
n11: OpenFile: key C0A80406:3411B44A:00000068:FFFFFFFF
n11:   status valid, token rf, addr 0x303CCDE8
n11:   lock state [ (no locks) ] flags [ ], writer 0xFFFFFFFF, wait shark
n11:   mutex 0x207D0138, cond 0x207D0188, eventP 0xC5AC408
n11:   inode number 104 nlink 1 genNum 1
n11:   metanode 192.168.14.1 (other node) token ro lock_state 0x00000000
n11:     takeovers 0 surrenders 0
n11: ....
n11: ....
```

Find which node is the Metadata Manager for file /chico/.profile by using the ls -il command. This gives you the i-node number of /chico/.profile. (In this case, it is 104.)

Then issue the mmfsadm dump files command and start scanning the output from the desired stripe group. The Metadata Manager for each file is located under the corresponding i-node number line in the output. In this case, the Metadata Manager for /chico/.profile is 192.168.14.1.

The Metadata Manager is selected to be the first node that had the file opened. It continues to provide metadata services for that file until one of the following events occurs:

- The file is closed everywhere.
- The node fails.
- The node resigns because it needs local resources.

When the Metadata Manager is down, the next node that needs the metadata service will become the Metadata Manager.

3.2.4 Token Manager Server Failure

Tokens are used to coordinate various activities occurring at the same time in the file system across nodes to make sure that the integrity of the file system is not compromised. They are used in much the same way that locks are used to coordinate the activities on a single node.

There is a Token Manager on every GPFS node. It runs completely in kernel mode and uses the kernel heap to store its tokens.

When a node wants to access data, it contacts the Token Manager Server to request a token. The Token Manager Server determines whether there is any locking conflict among the tokens that have already been granted and the currently requested one.

If there is no conflict, the Token Manager Server can allow the request to proceed by granting a token to that node, so that it can continue with what it wants to do. If there are conflicts, the Token Manager Server sends a list called a *copy set* that lists nodes that have conflict locks.

In order to reduce the workload at the Token Manager Server, it is the responsibility of the requesting Token Manager to negotiate with any node in the list to obtain the token.

There is one Token Manager Server per file system. It is located on the same node as the Stripe Group Manager. It is responsible for granting tokens to the requesting Token Managers.

For the purpose of availability and recoverability, two copies of the token are kept in the system: one in the Token Manager Server (the server copy), and one in the Token Manager (the client copy).

When a node is down, all the tokens it had can be recovered by obtaining the server copy from the Token Manager Server.

When the Token Manager Server is down, the new Token Manager Server can recover all the tokens that the old Token Manager Server had by obtaining the client copy from all Token Managers in the GPFS domain.

Chapter 4. Migration

This chapter discusses issues related to using data stored in a General Parallel File System(GPFS), including API interface considerations and performance-related issues.

4.1 Review of Various File Systems

The API and semantics of the BSD Fast File system considerably influenced UNIX and other similar file systems (for example, in the use of long file names, file sharing semantics, and so on). Over time, a variety of specialized file systems have been developed to extend the features of the BSD implementation.

However, these new features often come at a price. As an example, it might be possible to speed up some operations if metadata is not updated as it changes. Even in the BSD implementation, some operations do not cause some metadata to be updated, in order to save time.

Many file systems (especially those with special traits such as speed or network distribution) deviate from traditional local file systems based on the BSD model in some way. The following section describes how file systems have traded BSD compatibility for other capabilities.

4.1.1 Compatibility

Listed are some of the trade-offs that have been made in order to bring special features to file systems.

NFS In NFS there are a number of inconsistencies that can develop between different clients′ views of the file system. For example, metadata can be quite stale on an NFS client. Also, there is no requirement of synchronized times between clients and servers, so the *mtime*, *ctime* and *atimes* are difficult to use reliably. The precise order of many events cannot be determined in an NFS environment (for example, the O_EXCL is not supported on NFS). Applications must be prepared to handle more types of failures with NFS than with local file systems such as JFS. This is primarily a concern when NFS file systems are mounted with a *soft*, rather than a *hard*, NFS mount.

PIOFS The PIOFS file system conforms to many of the traditional local file system standards, but in this case compromises were made in order to allow PIOFS to provide better performance than traditional file systems. It is possible, under program control, to relax some consistency checks to improve performance. If applications are written to the PIOFS-specific interfaces, they can gain improved performance.

DFS DFS is a newer distributed file system than NFS. Many ideas from newer distributed file systems and lessons learned from problems with NFS have found their way into DFS. DFS clients are provided with file system semantics that very closely mimic those found in local file systems such as JFS.

DFS provides synchronization between reads and writes on the clients. This makes the programmer's job much easier and it is easier to develop correct systems. It is possible to relax this synchronization, but that requires explicit manipulation of the system by creating replicas. As with NFS, the DFS client can have different failure modes than are possible with a local file system such as JFS.

GPFS GPFS is based on technology that, like DFS, has learned from its predecessors. While DFS is designed to work well in a distributed environment, GPFS focuses on very high performance and use in a more limited distributed environment. GPFS also closely mimics local file system semantics, so it is relatively easy to use unaltered applications on GPFS file systems.

GPFS has one limitation (which is planned to be removed in a future version) and one exception.

The limitation is that at this time, GPFS files cannot be mapped into a process's address space using the shared/mapped memory facilities, but this restriction should be lifted in future versions.

The exception is the treatment of mtime, atime and ctime metadata. In the current implementation, metadata can be stale for up to 5 seconds.

JFS JFS is IBM's local file system that is used with AIX systems. JFS generally works the way a local file system developed based on the BSD file system is expected to work. JFS extends the BSD model by using a log to track metadata changes. This can significantly reduce the amount of time

needed to verify or repair a file system. This capability can come at the price of more head movement. However, placed in a system with more than one disk, the log can be placed on a different device and thus much better performance can be achieved.

4.1.2 Other Issues

Even if the APIs for two file systems are identical, there can be other differences in the way a file system performs. Different file systems behave in special ways when time or location are considered, and often a particular file system is chosen because of those special traits. Here are some examples:

NFS The distributed abilities and the large number of operating systems which contain client and server support are the keys to NFS success. Even though special code is needed to create robust systems using NFS, over time this code has become quite well understood, so this is not as severe a problem as might be expected.

PIOFS PIOFS is designed to allow special types of nonlinear access and very high data throughputs in SP environments. The *views* facility of PIOFS allows applications to access files in blocks that correspond to the data structures found in a program. For example, it is possible to read sequentially through rows or columns of an array stored in PIOFS.

There is a facility in PIOFS to checkpoint the version of a file. This allows a snapshot of a file to be saved for later use. Changes to a file are saved in new space on the disks so the old version can be retrieved. This is used to checkpoint files when they are in consistent states so a process can be restarted, perhaps after a failure. Also, an application could perform some transformation on the data, and later decide to go back to the previous version.

PIOFS can be configured to relax some file system semantics to provide better performance. This is a reasonable trade-off in many parallel applications since the work of parallelizing the code can be done in such a way that the application can guarantee that certain problem situations cannot happen.

DFS DFS provides the same semantics as most local file systems, like JFS. To reduce many performance costs, DFS caches data at the local machines. This reduces network traffic and, in most cases, allows the DFS client quicker access to the

data. There are cases where the costs associated with these robust semantics seem high. Still, there is great benefit in using unaltered applications and for many users, it is easier to live with these seldom-seen problems (which is only a problem of speed, not data corruption) in order to run software more easily.

GPFS In most cases, the operating system resources needed to provide a particular file system are small compared to those used by the applications. JFS uses very few system resources to provide a local file system. Distributed file systems such as NFS and DFS can consume more resources for network, memory, local caches and CPU needs. GPFS can consume a moderate amount of resources on the local machine. Usually this is a very good trade-off since the machine has little to do but wait for the data before it can complete its current task. However, in some cases, these additional resources might be more significant.

As Frederick Brooks wrote in *The Mythical Man Month*, "One has to ask, 'What does it do?' What does one get in ease-of-use and in performance for the dollars spent?" GPFS provides a very high performance capability with robust semantics. With GPFS, in most cases any use of system resources will be more than repaid in improved performance.

4.1.3 Access Control

A wide variety of access control models are found in the variety of file systems. If you now depend on special access control facilities from a file system, you will have to verify that the other file system you are considering utilizing will provide the same level of security.

In addition, if you are at a site that has special security requirements, refer to the specific details in the product documentation for the products used in your environment.

4.2 Migration to GPFS

Given all this background, how difficult is it to migrate to GPFS from some other file system? In most cases, there is nothing special to do because GPFS is a superset of the capabilities of most file systems you are currently using. Here are a few specific details:

NFS If an application works with NFS, it should have very little problem operating in a GPFS environment. One thing you should do, however, is to look for places where NFS allowed operations that would have been forbidden in most other file systems. An example of such an operation is having multiple processes create a file (*open(name, ... | O_EXCL | ... , mode)*) at the same time. This might work in NFS, but it will fail in GPFS. Also, at this time GPFS does not allow memory-mapped file operations.

JFS Except for memory-mapped operations and relaxed atime, ctime and mtime updates, most applications that operate correctly with JFS should run fine with GPFS. However, it is possible that some programs will behave differently with GPFS. As an example, a threaded program that relied on slower file system operations might now fail if GPFS is used, because it provides data faster than most other file systems.

DFS This is possibly the easiest migration since the applications should already be capable of operating correctly with a wide variation in I/O response times. About the only problem you may encounter is if your system relies on special features of DFS, such as enhanced access control lists or replication.

PIOFS Depending on what features of PIOFS you use, the migration effort can range from having to do nothing, to having it be impossible to perform a migration at all. In particular, PIOFS supports the use of *views* on the physical data. This facility allows an application to access the data in various strides. If your system depends on this facility, you will have to change your applications.

An intermediate-sized problem with the migration from PIOFS can occur if you made extensive use of the PIOFS tuning facilities. Under some conditions you can achieve better performance with the current PIOFS product. This can occur if you turn off the PIOFS checking for problems. At this time, GPFS does not allow this level of control.

Depending on access patterns, you may find that using GPFS replication, larger buffers, or network tuning will allow GPFS to achieve your performance needs. Again, any problems in this area will be limited to applications whose design is very closely tied to PIOFS.

In most cases with default installation parameters, GPFS should work as fast as, or faster, than PIOFS. In addition, GPFS does not have as many restrictions on how your application can use the file system. For example, PIOFS does not support locking, synchronous I/O (*O_SYNC*), non-blocking IO (*O_NONBLOCK*) and other flags such as *O_NDELAY, O_NOCTTY, O_NSHARE*, and so on. Finally, PIOFS has a few system-level problems such as not honoring resource limits, not allowing "holes" in files, and requiring that the root user can freely rsh between nodes.

There are other differences in the way PIOFS works. Generally, those features allow applications that are willing to put in file system-specific code to take advantage of special, high-performance paths to turn off time-consuming tests in the file system code. Most applications probably will not want to take on that extra level of responsibility, since the default performance of GPFS will be good enough.

In many systems PIOFS is used as a large "scratch" space and GPFS can work just fine as a replacement.

4.3 Migration Summary

In most cases it should be very easy to start to use GPFS in an existing system. GPFS currently has only one restriction (no shared memory operations), which should soon be lifted, and one design change (atime, ctime and mtime are not updated as often as on some other file systems).

If your application depends on a special characteristic of some other file system, or if it has quietly been taking advantage of some odd behavior, you may have to make changes.

As an easy test, if your application runs correctly on both JFS (robust semantics) and PIOFS (no memory-mapped file I/O), you can be relatively confident it will move to GPFS without change.

Chapter 5. Applications

This chapter describes the behavior of different applications when using the GPFS file system. We have included Lotus Notes as a commercial application since its use of files to manage databases is a good candidate for a parallel file system, especially if there is a need for scalability.

MPI applications are also discussed in this chapter, including both performance information and our test results when using GPFS as the file system of choice with these applications. The conclusions we present are based on the experience gathered when running a synthetic application using GPFS file systems. Appendix F, "How to Get the Examples in This Book" on page 213 explains how to obtain the code for this synthetic application.

Finally, we include a sort application written in C, which provided a good mix of reading and writing tasks that allowed us to draw some interesting conclusions regarding GPFS behavior in these kinds of environments.

5.1 Lotus Notes

In this section, we detail how to create and configure a GPFS environment in a RS/6000 SP system to achieve the best performance when running Lotus Domino Server Release 4.5. Note that all performance considerations described in 5.1.1, "GPFS Performance Considerations –" on page 82 can be applied to other applications having similar characteristics to Lotus Domino Server.

The purpose of this topic is to give information about how Lotus Domino server works on top of GPFS. We only worked with standalone Domino servers. All the recommendations and procedures we describe are intended for standalone Domino servers. If you are planning a multi-Domino server installation in your SP system, refer to the following URL: http://www.rs6000.ibm.com/resource/technology/NotesSPcfg Here yopu will will find a great deal of useful information about planning, installing, and configuring Lotus Domino servers on the RS/6000 SP.

In addition, information is provided about the Notes benchmarks we ran with different scenarios, and we describe the tool we used, the configurations of the scenarios, and the results we received.

Finally, we offer guidelines for migrating a current Lotus Domino Server Release 4.5 installation to a GPFS environment, and describe in which situations it is advisable to use GPFS for your Lotus Domino Server.

5.1.1 GPFS Performance Considerations –

Domino Server Rel. 4.5

You may ask, how can Lotus Domino Server benefit from GPFS?

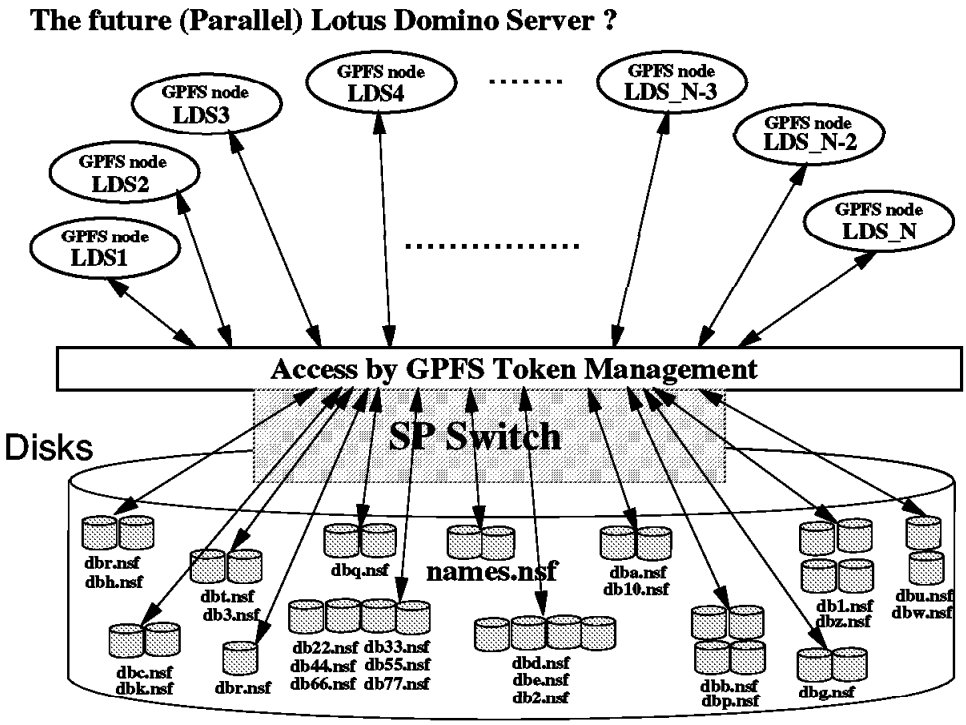


Figure 12. The Future, using GPFS

Using GPFS provides the following advantages, as illustrated in Figure 12.

- It gives transparent access to the Lotus Domino Server databases and/or program files. Each node of the GPFS cluster is able to access all Lotus Domino Server databases and program files that are located in GPFS file systems.

  Note: Unfortunately, at this time Lotus Notes is not able to use all the advantages GPFS offers since Lotus Domino Server Release 4.5

does not allow concurrent access to the Lotus Notes databases (the last version tested was 4.52). When a Lotus Domino Server accesses a Lotus Notes database, it locks the database/file.

However, you can change this behavior of the Lotus Domino Server by not locking any file but instead leaving this task to the GPFS locking mechanism. In this way, you can avoid having Lotus Notes replicas in the clusters of Lotus Notes Servers and reduce a great deal of CPU work and network traffic.

- It allows you to increase the aggregate bandwidth of the Lotus Domino Server file systems by spreading read/write operations across multiple disks of the GPFS cluster.
- It allows you to balance the workload across all disks to maximize their combined throughput.
- It allows you to support large amounts of data and have bigger file systems when necessary.
- It provides benefits when replicating data. GPFS allows you to have two copies of your data (even though your first choice should be to mirror your data). The data can be replicated at file system level or at file level.
- It provides system maintenance flexibility. You can add or delete disks when the file systems are mounted. You can also move your Lotus Domino Server from one node of the cluster to another with a minimum amount of manual intervention (if you have the necessary resources), when you need to perform maintenance tasks.

5.1.1.1 Performance Parameters

To achieve optimal performance from Lotus Domino Server running on top of GPFS, you must customize your GPFS environment. Later in this section, we present the best values we found for the GPFS parameters during our tests. These values can change for specific Lotus Notes configurations.

All these values were tested for standalone Lotus Notes Servers.

- Number of nodes (GPFS file system creation)

  This parameter is set at GPFS configuration time and *cannot* be changed later.

  The default value is 32 nodes.

  At the least, this parameter should be set at the maximum number of nodes you will have in the cluster. If you plan to have less than 32, leave the default value.

- Stripe Group Manager

  According to our tests, you can run the Stripe Group Manager in the Application server node or in a VSD server node. Nevertheless, you should check the CPU load.

  You can force it by using the /var/mmfs/etc/cluster.preferences file.

  Attention

  For more information about how to determine which node is the Stripe Group Manager for a particular GPFS file system and how to force it to be in a specific GPFS node of the cluster, refer to 3.2.2, “Stripe Group Manager Failure” on page 69.

- Disk Planning (prior to GPFS configuration)

  Before you install and configure GPFS, you must carefully plan which disks will provide the best solution for your needs. Follow the general recommendations for GPFS:

  - Use dedicated VSD Server nodes. We recommend that you do not run your Lotus Domino Server on VSD Server nodes.
  - You should have one VSD on one disk.
  - Balance the disks; that is, try to have the same number of disks on each node.
  - SSA disks should be the default solution.
  - Use twin-tailing for improved availability.

- Stripe method (GPFS file system creation)

  The default value is round robin and it should be the normally selected method.

  Note: This value can be changed after file system creation.

- File System block size (GPFS file system creation)

  For Lotus Domino Server, use a block size of 64KB, which is the medium block size available in GPFS file systems.

  Using a block size of 16KB will result in more efficient use of disk space, but will provide worse performance.

  Using a block size of 256KB wastes disk space and does not provide better performance.

  Note: This value *cannot* be changed after file system creation.

- I-node (GPFS file system creation)

This is not a determinant parameter. A 1KB i-node size value should be correct.

Note: This value *cannot* be changed after file system creation.

- Indirect blocks

  This is not a determinant parameter.

  For the Lotus Domino Server, use an indirect block size of 8KB.

  Note: This value *cannot* be changed after file system creation.

- Data replication (GPFS file system creation)

  Data replication is an excellent high availability solution having easy and flexible management. It is recommended for mostly read configurations, since it will improve performance.

  Replication can be done at file system level or at file level, and you can have up to two data replicas of your file or file systems.

  Note: This value can be changed after file system creation (only of max. number of replicas is set to two at file system creation time).

- Metadata replication (GPFS file system creation)

  Metadata replication is related to data replication; that is, if you decide to replicate some data, you should also replicate the metadata.

  Replication can be done at file system level or at file level, and you can have up to two metadata replicas of your files or your file systems.

  Note: This value can be changed after file system creation (only of max. number of replicas is set to two at file system creation time).

- maxFilesToCache (GPFS initial configuration)

  This refers to the maximum number of i-nodes to cache. This parameter is an internal GPFS parameter and cannot be modified with the standard GPFS tools. Refer to Appendix C, “The GPFS Configuration File (mmsdrcfg1)” on page 191 for details about how to change this value.

  This value must be the maximum number of concurrently accessed files that you will have in your GPFS file systems. This parameter can have *different values* on *different nodes*. It is very important that you know the behavior of your applications in each node in order to set the value for this parameter as correctly as possible in each node.

Note: Use the shareware tool described in Appendix F, “How to Get the Examples in This Book” on page 213 to calculate the maximum number of concurrently accessed files in the GPFS file systems of each node.

To activate the changes of this parameter, you must restart GPFS on the affected node.

- pagepool (GPFS initial configuration)

  This refers to the data cache on each node. The default value is 20MB.

  This value should be increased. The pagepool is directly related to the `maxFileToCache` GPFS parameter. You should try to have a pagepool of:

  Pagepool

  ```
  pagepool = BlockSize * Number_of_concurrent_files * 2
  ```

  `Number_of_concurrent_files` is the number of files concurrently accessed in all the GPFS file systems of each particular node.

  Refer to Appendix C, “The GPFS Configuration File (mmsdrcfg1)” on page 191 for details about how to change this value.

  Important

  The value of this parameter will be pinned memory on the GPFS node. If you cannot dedicate the necessary memory in your node to satisfy the formula, you can set a lower pagepool value. However, try to be as close as possible to the formula value.

  To activate the changes of this parameter, you must restart GPFS on the affected node.

- mallocsize (GPFS initial configuration)

  This refers to the metadata cache on each node. The default value is 4MB.

  This value must be increased by following the rule:

  Mallocsize

  ```
  mallocsize = 4KB * maxFilesToCache + 2MB
  ```

  Refer to Appendix C, “The GPFS Configuration File (mmsdrcfg1)” on page 191 for details about how to change this value.

Important

The value of this parameter will be pinned memory on the GPFS node. If you cannot dedicate the necessary memory in your node to satisfy the formula, you can set a lower mallocsize value. However, try to be as close as possible to the formula value.

To activate the changes of this parameter, you must restart GPFS on the affected node.

- Priority of daemons (GPFS initial configuration)

  We recommend that you *do not* change the priority of GPFS daemons. Trying to improve the performance of GPFS by increasing the priority of GPFS daemons can generate unexpected results in the system. The default should be 40.

  To activate the changes of this parameter, you must restart GPFS.

5.1.2 Lotus Domino Server on a RS/6000 SP Node Using GPFS

In order to be able to go through the steps explained in this topic, a good working knowledge of RS/6000 SP systems, POWERparallel System Support Programs, and Lotus Domino Server is needed.

The installation process of Lotus Domino Server Release 4.5 is exactly the same as it is in a standalone AIX system. The only differences are in the creation and configuration processes of the file systems where you will place the databases of Lotus Domino Server. Once you have the file systems up and running, the remaining steps to install the program files and later the databases are the same as with a standalone AIX system.

In our implementation, we will install the binaries of Lotus Domino Server in a local JFS file system and the Domino Server databases in a GPFS file system. With this configuration, we will be able to start the Lotus Domino Server in each node where all the GPFS file systems containing the Domino Server databases are mounted, and which have locally installed the Lotus Domino Server binaries.

Another possible option would be to install the Domino Server binaries in a GPFS file system. With this option, you would be able to run the Lotus Domino Server in each node of the GPFS cluster with only one installation of the binaries for the entire cluster.

However, at this time we recommend you *do not* install the Domino Server binaries in a GPFS file system because of the impact upon performance.

Important

Remember that once the GPFS file system is created, every node of that GPFS cluster can mount it locally.

5.1.2.1 Creating the AIX JFS File System for the Domino Server Binaries

1. Create a new volume group for Domino Server binaries.

   If you do not want to create a new volume group for the binaries, you can skip this step and go to “Create the File System for the Domino Server Binaries” on page 88.

   Log on to the node where you will run Lotus Domino Server as user root.

   Type in the following command:

```
sp21n15:/> smitty mkvg
```

   The next screen will be displayed:

```
                              Add a Volume Group

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  VOLUME GROUP name                                  [notesvg]
  Physical partition SIZE in megabytes                4                      +
* PHYSICAL VOLUME names                              [hdisk2]       +
  Activate volume group AUTOMATICALLY                 yes                    +
    at system restart?
  Volume group MAJOR NUMBER                          []                      +#
  Create VG Concurrent Capable?                       yes                    +
  Auto-varyon in Concurrent Mode?                     yes                    +









F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

   In this example, the volume group is called notesvg and will be created in hdisk2.

2. Create the file system for the Domino Server binaries.

   The file system for the Domino Server binaries will be /usr/lpp/lotus, with a size of 150MB.

   As root, type the following command:

```
sp21n15:/> smitty crjfs
```

   The Add a Journaled File System screen will be displayed.

```
                          Add a Journaled File System

Move cursor to desired item and press Enter.

  Add a Standard Journaled File System
  Add a Compressed Journaled File System
  Add a Large File Enabled Journaled File System









F1=Help             F2=Refresh          F3=Cancel           F8=Image
F9=Shell            F10=Exit            Enter=Do
```

   When you select **Add a Standard Journaled File System** by pressing the Enter key, the Volume Group Name screen will be displayed:

```
                            Volume Group Name

 Move cursor to desired item and press Enter.

   rootvg
   notesvg

 F1=Help                F2=Refresh             F3=Cancel
 F8=Image               F10=Exit               Enter=Do
 /=Find                 n=Find Next
```

   Select the volume group where you want to place the file system, for example **notesvg**, and enter the parameters on the Add a Standard Journaled File System screen:

```
                    Add a Standard Journaled File System

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Volume group name                                   notesvg
* SIZE of file system (in 512-byte blocks)           [314400]       #
* MOUNT POINT                                        [/usr/lpp/lotus]
  Mount AUTOMATICALLY at system restart?              yes          +
  PERMISSIONS                                         read/write              +
  Mount OPTIONS                                      []                       +
  Start Disk Accounting?                              no                      +
  Fragment Size (bytes)                               4096                    +
  Number of bytes per inode                           4096                    +
  Allocation Group Size (MBytes)                      8                       +

F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Now you can mount the new file system and install the Domino Server binaries, or you can continue with the creation of the GPFS file system for the Domino Server databases.

5.1.2.2 Creating the GPFS File System for the Domino Server Databases

In this section, we provide an example of a recommended environment configuration to help you achieve the best results when running your Lotus Domino Server.

The hardware configuration for this example is:

- Application server node:
  - High node
  - Six PowerPC 604 processors (112 MHz)
  - 512MB RAM
- VSD server node:
  - Two wide nodes
  - POWER2, 66MHz
  - 256MB RAM
- Disks:
  - Two loops of four 4.5GB SSA disks shared between the two VSD server nodes (using only three disks of each loop)

Before creating the GPFS file system, you should be aware of the following performance considerations:

- The Lotus Domino Server will run in a non-VSD server node in order to avoid spending time in I/O operations in the Application server node.
- Each 4.5GB SSA disk will contain only one VSD.
- The I/O load will be completely balanced between the two VSD server nodes. Each VSD server node will be the VSD server primary node for three VSDs.
- The three VSDs/disks controlled by each VSD server primary node will be in a different loop, in order to separate the data traffic in the loops.
- The binaries will be installed in the local JFS file system of the Application server node.

Creation of the GPFS File System

1. To create the VSDs for the GPFS file system:

   Important

   - Be sure GPFS is up and running in the cluster.
   - Be sure you have all the hardware resources connected and available on the systems where you will work.
   - Before you create the VSDs, you must know exactly which disks of which nodes you will use for the GPFS file system. You must also decide which VSD server node will be the VSD server primary node for each VSD.
   - In our example, we follow the previously mentioned considerations in order to balance the I/O load as much as possible between the VSD server nodes.
   - You can use the new SMIT panels for SSA disks to determine which loop each SSA disk belongs to, and use the maymap tool to provide a map of the SSA loops, along with the disk ID of each SSA disk. For more information about the maymap tool, refer to Appendix D, “SSA Configuration” on page 197.

   Log on to the Control Workstation as user root.

   Type in the following command:

```
sp21en0:/> smitty createvsd_dialog
```

Enter the necessary parameters in the Create a Virtual Shared Disk screen:

```
                        Create a Virtual Shared Disk

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Node and Disk Information                        [13/11:hdisk2/]
* Logical Volume Group Name                        [lot1]
* Virtual Shared Disk size in Mega Bytes           [4296]                  #
  Virtual Shared Disk Option                       [nocache]              +
  Number of Virtual Shared Disks per Node          []
  One Virtual Shared Disk per Node: not cyclic     [ ]                    +
  Virtual Shared Disk Name Prefix                  [lot1vsd]
  Logical Volume Name Prefix                       [lot1]
  Logical Volume Partition Size                    [8]                    +#
  Logical Volume Mirror Count                      [1]                    +#
  Logical Volume Stripe Size                       [ ]                    +#




F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

In this example, we create the first VSD (out of a total of six) that we will use in our configuration.

As you can see, the VSD server primary node for this VSD is sp21n13 and the VSD server secondary node is sp21n11.

The next two VSDs are created in the same way; we only changed the VG, VSD and LV names, and the name of the disk.

In the next Create a Virtual Shared Disk screen, we create the first VSD for the other VSD server primary node, as shown:

```
                        Create a Virtual Shared Disk

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                    [Entry Fields]
* Node and Disk Information                        [11/13:hdisk9/]
* Logical Volume Group Name                        [lot5]
* Virtual Shared Disk size in Mega Bytes           [4296]                  #
  Virtual Shared Disk Option                       [nocache]              +
  Number of Virtual Shared Disks per Node          []
  One Virtual Shared Disk per Node: not cyclic     [ ]                    +
  Virtual Shared Disk Name Prefix                  [lot5vsd]
  Logical Volume Name Prefix                       [lot5]
  Logical Volume Partition Size                    [8]                    +#
  Logical Volume Mirror Count                      [1]                    +#
  Logical Volume Stripe Size                       [ ]                    +#




F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

As you can see, the VSD server primary node for this VSD is sp21n11 and the VSD server secondary node is sp21n13.

The next two VSDs are created in the same way; we only changed the VG, VSD and LV names, and the name of the disk.

The following screen shows the results of the creation of the six VSDs:

```
VSD server nodes:
sp21n11
sp21n13
VG Information:
Command:
sp21en0:/> dsh -w sp21n11 lsvg -o | grep lot
sp21n11: lot7
sp21n11: lot6
sp21n11: lot5
Command:
sp21en0:/> dsh -w sp21n13 lsvg -o | grep lot
sp21n13: lot3
sp21n13: lot2
sp21n13: lot1

LV Information:
Command:
sp21en0:/> dsh -w sp21n11 lsvg -l lot5
sp21n11: lot51n11            jfs        537   537   1   closed/syncd   N/A
Command:
sp21en0:/> dsh -w sp21n11 lsvg -l lot6
sp21n11: lot61n11            jfs        537   537   1   closed/syncd   N/A
Command:
sp21en0:/> dsh -w sp21n11 lsvg -l lot7
sp21n11: lot71n11            jfs        537   537   1   closed/syncd   N/A
Command:
sp21en0:/> dsh -w sp21n13 lsvg -l lot1
sp21n13: lot11n13            jfs        537   537   1   closed/syncd   N/A
Command:
sp21en0:/> dsh -w sp21n13 lsvg -l lot2
sp21n13: lot21n13            jfs        537   537   1   closed/syncd   N/A
Command:
sp21en0:/> dsh -w sp21n13 lsvg -l lot3
sp21n13: lot31n13            jfs        537   537   1   closed/syncd   N/A
```

2. Configure the VSDs in all nodes.

   Now you must configure and start the six VSDs in order to use them later to create the GPFS file system. From the Control Workstation, issue the following command:

```
sp21en0:/> dsh /usr/lpp/csd/bin/cfgvsd 'lot5vsd1n11 lot6vsd1n11
lot7vsd1n11 lot1vsd1n13 lot2vsd1n13 lot3vsd1n13'
```

3. Start the VSDs in all nodes.

   Once the VSDs are configured (the previous command put these VSDs in STP state), you can start them. From the Control Workstation, issue the command:

```
sp21en0:/> dsh /usr/lpp/csd/bin/startvsd 'lot5vsd1n11 lot6vsd1n11
lot7vsd1n11 lot1vsd1n13 lot2vsd1n13 lot3vsd1n13'
```

4. Check the VSDs.

Before you create the GPFS file system, check the state of the VSDs by running the following command from the Control Workstation:

```
sp21en0:/> dsh /usr/lpp/csd/bin/lsvsd -l | grep lot
```

You should see the following screen, with all VSDs in an active (ACT) state:

```
(previous nodes)

sp21n13: 11      ACT   13      36       1      lot1vsd1n13
nocache       4296
sp21n13: 12      ACT   13      37       1      lot2vsd1n13
nocache       4296
sp21n13: 13      ACT   13      39       1      lot3vsd1n13
nocache       4296
sp21n13: 16      ACT   11       0       0      lot5vsd1n11
nocache       4296
sp21n13: 17      ACT   11       0       0      lot6vsd1n11
nocache       4296
sp21n13: 18      ACT   11       0       0      lot7vsd1n11
nocache       4296

(next nodes)
```

5. Create the GPFS file system.

   As with the VSD configuration, we follow the performance considerations explained in 5.1.1, “GPFS Performance Considerations –” on page 82.

   First we create the *description file* for the new GPFS file system.

   Attention

   For more information about creating description files, refer to 2.2.2.3, “Starting Your GPFS Daemon” on page 44 and to *General Parallel File System for AIX: Installation and Administration*.

   Our description file /tmp/gpfs.lotus contains the following:

```
lot1vsd1n13::::
lot2vsd1n13::::
lot3vsd1n13::::
lot5vsd1n11::::
lot6vsd1n11::::
lot7vsd1n11::::
```

   As user root, log on to the node of the GPFS cluster where the description file /tmp/gpfs.lotus is located.

   Issue the following command:

```
sp21n13:/> /usr/lpp/mmfs/bin/mmcrfs /home/notes lotus -F /tmp/gpfs.lotus
-A yes -B 64K -i 1K -I 8K
```

In our example:

- The mount point will be /home/notes.
- The device name will be lotus.
- The system device file name will be /dev/lotus.
- The DescFile will be /tmp/gpfs.lotus.
- We will mount the file system automatically on the GPFS start.
- The block size will be 64KB.
- The indirect size will be 8KB.
- The i-node size will be 1KB.

Figure 13 shows the new GPFS file system:

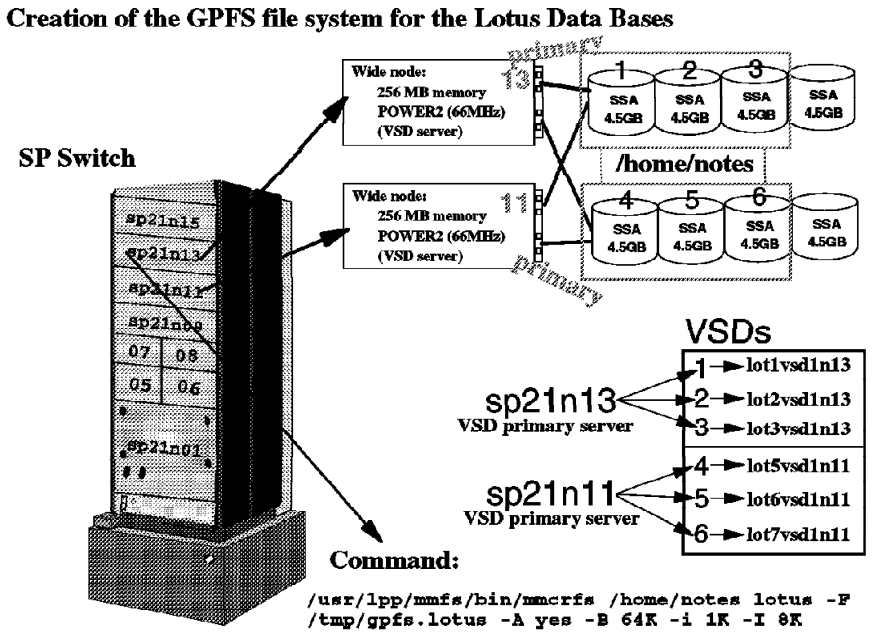


Figure 13. New GPFS File System. This figure graphically illustrates all the system resources involved in the creation of the GPFS file system.

If the command completes correctly, you will get the following screen:

```
mmcrfs: 6027-1052 Making MMFS filesystem /dev/lotus
/usr/sbin/tscrfs /dev/lotus -F /tmp/mmcrfsdddo22714 -I 1024 -M 1 -n 32 -R 1 -s
oundRobin -v no -c 0 -f 512 -p no -w 0

MMFS: 6027-531 The following disks of lotus will be formatted:
    lot1vsd1n13: size 4399104KB
    lot2vsd1n13: size 4399104KB
    lot3vsd1n13: size 4399104KB
    lot5vsd1n11: size 4399104KB
    lot6vsd1n11: size 4399104KB
    lot7vsd1n11: size 4399104KB
MMFS: 6027-540 Formatting file system ...
MMFS: 6027-572 Completed creation of file system /dev/lotus.
```

6. Mount the GPFS file system.

   If you want to mount the new GPFS file system /home/notes on all nodes of the GPFS cluster, log on to the Control Workstation as user root and execute the following command:

```
sp21en0:/> dsh mount /home/notes
```

7. Check the GPFS file system.

   Log on to one node of the GPFS cluster and type `mmdf lotus`. You will get the next screen:

```
disk            disk size  failure holds    holds    in full        in
name                in KB    group metadata data      blocks fragments
--------------- --------- -------- -------- ------ --------- ---------
lot7vsd1n11       4399104     4011 yes      yes      3713488      2086
lot6vsd1n11       4399104     4011 yes      yes      3714256      1926
lot5vsd1n11       4399104     4011 yes      yes      3713280      1840
lot3vsd1n13       4399104     4013 yes      yes      3713424      2008
lot2vsd1n13       4399104     4013 yes      yes      3713456      2057
lot1vsd1n13       4399104     4013 yes      yes      3713264      2240
                 ---------                          --------- ---------
(total)          26394627                            22281168     12159
```

Next, you should configure the user environment.

The installation of the Domino Server binaries can be done just after the creation of the local JFS. However, the installation of the Domino Server databases must be done *after* the configuration of the user environment.

5.1.2.3 Create and Configure the User Environment

When installing a RS/6000 SP system, you can choose between two options to manage the users in your system:

- Use PSSP user management *with* AIX automounter and File Collections.

- Not using PSSP user management or using it without AIX Automounter and File Collections.

For other configurations, it will be necessary to adapt some of the steps explained here.

To determine whether or not you are using PSSP user management with AIX automounter and File Collections, issue the following command on the Control Workstation:

```
sp21en0:/> splstdata -e
```

You will see the following screen if you are using PSSP user management with AIX automounter and File Collections:

```
             List Site Environment Database Information

attribute               value
------------------------------------------------------------------------------
control_workstation     sp21en0
cw_ipaddrs              9.12.1.137:192.168.4.137:
install_image           bos.obj.ssp.421
remove_image            false
primary_node            1
ntp_config              consensus
ntp_server              ""
ntp_version             3
amd_config              true
print_config            false
print_id                ""
usermgmt_config         true
passwd_file             /etc/passwd
passwd_file_loc         sp21en0
homedir_server          sp21en0
homedir_path            /home/sp21en0
filecoll_config         true
supman_uid              102
supfilesrv_port         8431
spacct_enable           false
spacct_actnode_thresh   80
spacct_excluse_enable   false
acct_master             0
cw_has_usr_clients      false
code_version            PSSP-2.3
layout_dir              ""
authent_server          ssp
backup_cw               ""
ipaddrs_bucw            ""
active_cw               ""
cw_lppsource_name       aix421
```

If you are not using AIX automounter and File Collections, you will see the following screen:

```
             List Site Environment Database Information

attribute               value
------------------------------------------------------------------------------
control_workstation     sp2en0
cw_ipaddrs              192.168.3.37:192.168.3.38:9.12.1.37
install_image           bos.obj.ssp.421
remove_image            false
primary_node            1
ntp_config              consensus
ntp_server              ""
ntp_version             3
amd_config              false
print_config            false
print_id                ""
usermgmt_config         false
passwd_file             /etc/passwd
passwd_file_loc         sp2en0
homedir_server          sp2en0
homedir_path            /home/sp2en0
filecoll_config         false
supman_uid              102
supfilesrv_port         8431
spacct_enable           false
spacct_actnode_thresh   80
spacct_excluse_enable   false
acct_master             0
cw_has_usr_clients      false
code_version            PSSP-2.3
layout_dir              /spdata/sys1/syspar_configs/1nsb0isb/config.4_12/layout.3
authent_server          ssp
backup_cw               ""
ipaddrs_bucw            ""
active_cw               ""
cw_lppsource_name       aix421
```

The **usermgmt_config** option is set to **false**, reflecting that we are not using the PSSP user management facilities.

Notes:

1. When using PSSP user management with File Collections, remember that the *supper* process updates the File Collections every hour, thus overwriting any local modification on the nodes. All changes related to the File Collections done in the Control Workstation will be updated every hour in the nodes.
2. The AIX automounter, transparently, is responsible for mounting each remote user directory in the machine you are logged in to, but it is your responsibility to allow all nodes to access the remote directory NFS by exporting the remote directory on the remote node.

There are important considerations to keep in mind when using the AIX automounter; these considerations are explained in 5.1.4, “Moving the Lotus Domino Server between RS/6000 SP Nodes” on page 117.

Using PSSP user management with AIX automounter and File Collections:

If you are using PSSP user management *with* AIX automounter and File Collections, you should follow the next steps to create a correct environment for the Lotus Notes users.

Important

If you want to have more than one Lotus Notes user in order to run different workstations or different servers, you should repeat these steps for *each* new Lotus Notes user.

1. Create the notes group in the Control Workstation.

   Log on to the Control Workstation as user root.

   Type:

   ```
   sp21en0:/> mkgroup - 'A' notes
   ```

2. Create the Lotus Notes user in the Control Workstation.

   Attention

   The home directory of the notes user should be on the GPFS file system previously created for the Domino Server databases.

   Log on to the Control Workstation as user root.

   Type:

   ```
   sp21en0:/> smitty spmkuser
   ```

   Enter the parameters on the Add a User screen.

```
                                  Add a User

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* User NAME                                          [notes]
  User ID                                            []                          #
  LOGIN user?                                         true                       +
  PRIMARY group                                      [notes]                     +
  Secondary GROUPS                                   []                          +
  HOME directory                                     [sp21n15:/home/notes]
  Initial PROGRAM                                    []                          /
  User INFORMATION                                   [Lotus Notes User]
```

After the execution of the command, the next screen will appear:

```
                                COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

spmkuser:  0027-158 Attention:  the directory /home/notes on sp21n11 already
     exists but has different ownership than the id of the user to be created.
     Modify the ownership if that directory does not belong to another user.



F1=Help             F2=Refresh          F3=Cancel           F6=Command
F8=Image            F9=Shell            F10=Exit            /=Find
n=Find Next
```

This warning is normal because on the remote node the new user is not set up.

3. Export the remote directory on the remote node.

   Log on as user root to the remote node where the home directory of the new user is located.

   Type:

```
sp21n15:/> /usr/sbin/mknfsexp -d '/home' -t 'rw' '-B'
```

4. Set the password for the new user.

   Log on to the Control Workstation as user root.

   Type:

```
sp21en0:/> smitty passwd
```

   Enter the user name parameter on the Change a User's Password screen and press Enter.

```
                    Change a User's Password

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  User NAME                                          [notes]       +



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

On the next screen, enter the password for the user notes.

```
Changing password for "notes"
notes's New password:
```

Re-enter the same password.

```
Changing password for "notes"
notes's New password:
Enter the new password again:
```

Exit SMIT.

5. Log on the first time.

   To avoid having to change the user password more than once when logging in, we perform the first login from the Control Workstation.

   Log on to the Control Workstation as user notes.

   After you enter notes's password, the system will ask you for a new password, as shown:

```
AIX Version 4
(C) Copyrights by IBM and by others 1982, 1996.
login: notes
notes's Password:
3004-610 You are required to change your password.
        Please choose a new one.

notes's New password:
```

Enter the password, and then re-enter the same password.

```
AIX Version 4
(C) Copyrights by IBM and by others 1982, 1996.
login: notes
notes's Password:
3004-610 You are required to change your password.
        Please choose a new one.

notes's New password:
Enter the new password again:
```

6. Propagate all the information of the new user.

   In order to propagate the group and user information to all the RS/6000 SP nodes as quickly as possible, you must update the user.admin File Collection in all of them. The *supper* process does this update every hour. To force the update of this File Collection, you must do the following:

   Log on to the Control Workstation as user root.

   Type:

   ```
   sp21en0:/> dsh /var/sysman/supper update user.admin
   ```

   Note: You must set up your user root environment correctly to ensure this command will execute correctly.

7. Set the correct owner/group and permissions on the remote node.

   After the group and user information is updated on all the RS/6000 SP nodes, you can set up correctly the owner, group and file permissions for the home directory of the new user in the remote

node. If the home directory of the user is the mount point of a file system, you should first mount the file system.

Log on to the remote node as user root.

Type:

```
sp21n15:/> mount /home/notes
```

After the GPFS file system is mounted, type:

```
sp21n15:/> chown notes.notes /home/notes
```

You should be able to use the default permissions of the directory.

Using PSSP user management without AIX automounter and File Collections:

If you are not using PSSP user management with AIX automounter and File Collections, you should set up the new group and the new user locally in the RS/6000 SP node where you want to run the Lotus Domino Server. In this case, the process is exactly the same as in a standalone AIX environment.

With this configuration, the Lotus Domino Server can only run on the RS/6000 SP node where you created and configured the user environment.

5.1.2.4 Installing the Domino Server binaries

This procedure is fully explained in *Lotus Notes Release 4.5 on AIX Systems: Installation, Customization and Administration*.

5.1.2.5 Installing the Domino Server databases

This procedure is fully explained in *Lotus Notes Release 4.5 on AIX Systems: Installation, Customization and Administration*.

Note: Remember that the PATH for this installation must be /home/notes/notesr4, and it must be in the same node where you installed the Domino Server binaries.

5.1.3 Comparison of JFS and GPFS Environments

This section provides the information we gathered from the benchmarks which were run over different configurations: local JFS configurations and GPFS configurations.

We also ran the same benchmarks for different Application server node hardware configurations.

Attention

The tool we used to do these benchmarks is Lotus NotesBench for Lotus Notes R4.

The benchmark we used in our tests is the Mail and Shared Database.

The configuration of Lotus NotesBench for Lotus Notes R4 was exactly the same for all the scenarios.

Disclaimer

In regard to Lotus NotesBench for Lotus Notes R4:

We choose Lotus NotesBench for Lotus Notes R4 because it is the standard tool for Lotus Notes benchmarking.

The personnel who installed, configured, and ran the tool were not certified.

The setup of the benchmarks was done by strictly following the guidelines given in the NotesBench documentation.

5.1.3.1 Summary of the Tests

We set up and ran Mail and Shared Database benchmarks because we consider this to be the more common use of Lotus Domino Server. Due to time restrictions, we were not able to run other benchmarks such as Groupware A or the Shared Discussion database.

At the end of this section, we provide detailed conclusions about all tests and results. These conclusions and results can be used as starting points for applications which behave like Lotus Notes.

5.1.3.2 Objectives

The objectives of the benchmarks were:

- To show the advantages of running Lotus Domino Server on top of GPFS.
- To describe the best hardware configuration we found for that purpose.

- To provide you with the optimal values for all GPFS parameters so you can achieve the best results when running Lotus Domino Server on top of GPFS.
- To provide you with additional details about the behavior of GPFS and Lotus Domino Server running together.
- To determine when it is advisable to use GPFS as the file system support for Lotus Domino Server, instead of using local JFS.
- To extract as much useful information as possible from the benchmarks.

5.1.3.3 Hardware and Software Configuration of the Test Scenarios

This section lists the hardware and software used for the benchmarks

First we describe the system drivers, which were the same for all the benchmarks, without any changes.

Note: We found that the wide nodes were the best I/O performers.

System Drivers

1. Hardware
   - Eight thin nodes with one POWER2 processor (66 MHz). The parent shared one node with one child.
   - 128MB RAM.
   - Ethernet (10MB/s).
2. Software
   - AIX 4.2.1.
   - Lotus Notes Workstation Release 4.5a.
   - The Domino Server binaries were installed in a file system using internal SCSI disks.
   - There was a common NFS directory for the child and result files of each child and the parent.

Following are the different Server Under Test configurations used during all our tests:

1. First scenario: **Local JFS configuration with high node**
   a. Hardware
      - High node with six PowerPC 604 processors (112 MHz)

      Application server node and I/O server:
      - 512MB RAM
      - One SSA adapter

- Two loops of four 4.5GB SSA disks

b. Software

- AIX 4.2.1
- PSSP 2.4
- Lotus Notes Domino Server Release 4.5a.
- The Domino Server binaries were installed in a file system in internal SCSI disks.
- The Domino Server databases were installed in two filesystems in external SSA disks, in different SSA loops.

Figure 14 illustrates the first scenario.

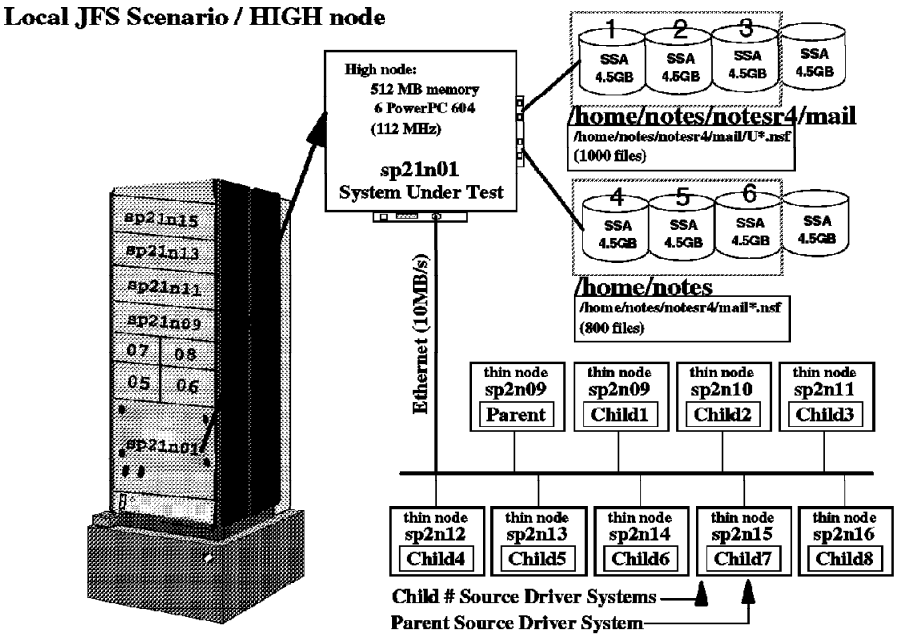


Figure 14. Local JFS Scenario - High Node. High RS/6000 SP node with 512MB of RAM, six PowerPC 604 processors, and two loops of four 4.5GB SSA disks in one SSA adapter.

2. Second scenario: **Local JFS configuration with wide node**

a. Hardware

- Wide node with one POWER2 processor (66 MHz)

Application server node and I/O server:

- 512MB RAM
- One SSA adapter

- Two loops of four 4.5GB SSA disks

b. Software

- AIX 4.2.1
- PSSP 2.4
- Lotus Notes Domino Server Release 4.5a.
- The Domino Server binaries were installed in a file system in internal SCSI disks.
- The Domino Server databases were installed in two filesystems in external SSA disks, in different SSA loops.

Figure 15 illustrates the second scenario.

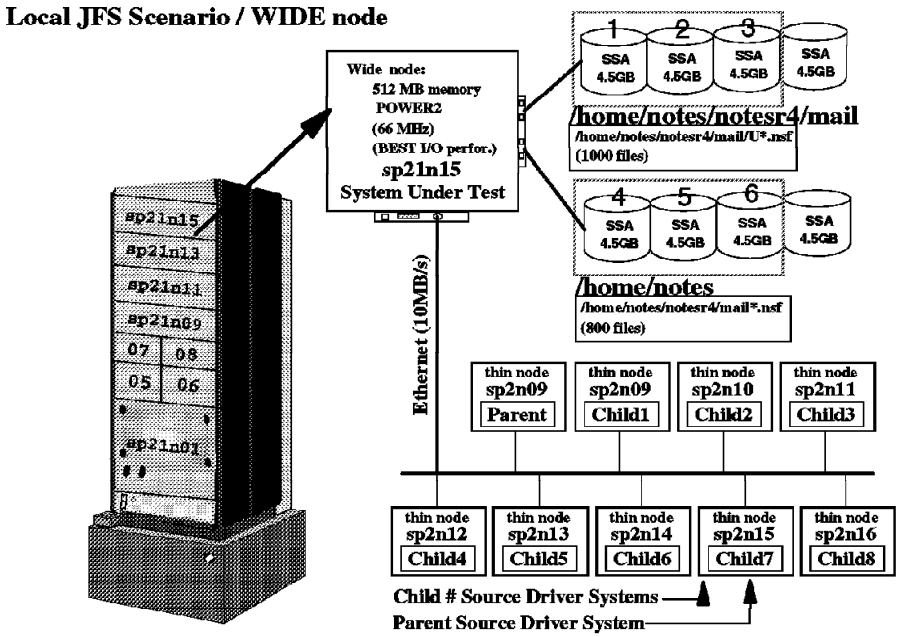


Figure 15. Local JFS Scenario - Wide Node. Wide RS/6000 SP node with 512MB of RAM, one POWER2 processor, and two loops of four 4.5GB SSA disks in one SSA adapter.

3. Third scenario: **GPFS configuration with high node**

a. Hardware

- High node with six PowerPC 604 processors
- 512MB RAM

Application server node:

- Two wide nodes with one POWER2 processor (66 MHz)

VSD server nodes

256MB RAM on each one

One SSA adapter on each one

Two loops of four 4.5GB SSA disks shared between the two VSD server nodes

b. Software

- AIX 4.2.1
- PSSP 2.4
- RVSD 2.1.1
- GPFS 1.1
- Lotus Notes Domino Server Release 4.5a.
- The Domino Server binaries were installed in a file system in internal SCSI disks.
- The Domino Server databases were installed in one GPFS file system spread in six SSA disks. Each disk contained one VSD. Each VSD server node was VSD server primary node for three VSDs, and the three SSA disks of each VSD server primary node were in a separate loop.

Figure 16 on page 110 illustrates the third scenario.

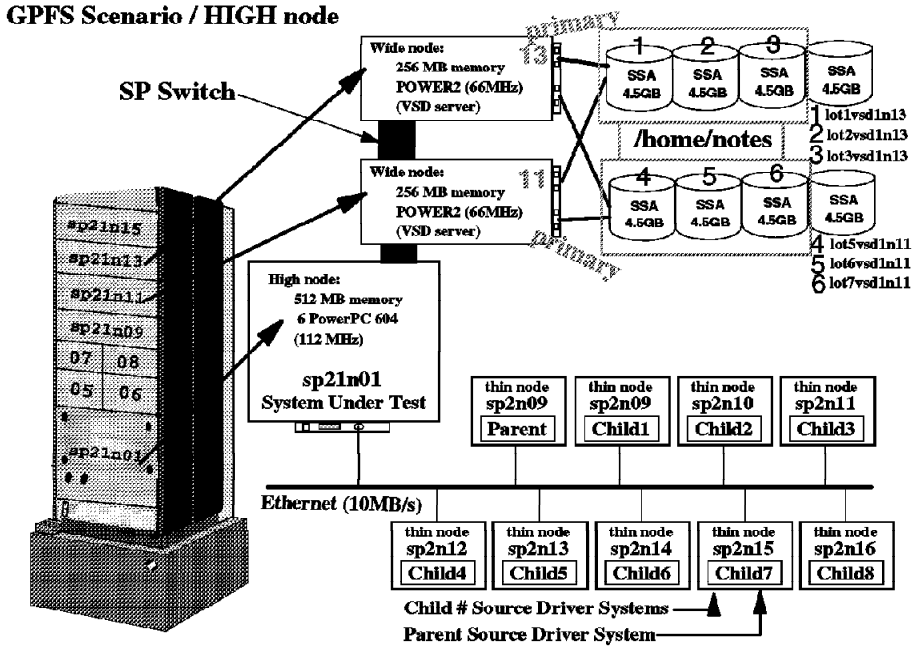


Figure 16. GPFS High Node Scenario. High RS/6000 SP node with 512MB of RAM and six PowerPC 604 processors as Application server node. Two wide RS/6000 SP nodes with 256MB of RAM and one POWER2 processor as VSD servers. Two loops of four 4.5GB SSA disks shared between the two VSD server nodes.

4. Fourth scenario: **GPFS configuration with wide node**

   a. Hardware

   - Wide node with one POWER2 processor (66 MHz)
   - 512MB RAM

   Application server node:

   - Two wide nodes with one POWER2 processor (66 MHz)

   VSD server nodes

   256MB RAM on each one

   One SSA adapter on each one

   Two loops of four 4.5GB SSA disks shared between the two VSD server nodes

   b. Software

   - AIX 4.2.1
   - PSSP 2.4
   - RVSD 2.1.1

- GPFS 1.1
- Lotus Notes Domino Server Release 4.5a.
- The Domino Server binaries were installed in a file system in internal SCSI disks.
- The Domino Server databases were installed in one GPFS file system spread in six SSA disks. Each disk contained one VSD. Each VSD server node was VSD server primary node for three VSDs, and the three SSA disks of each VSD server primary node were in a separate loop.

Figure 17 illustrates the fourth scenario.

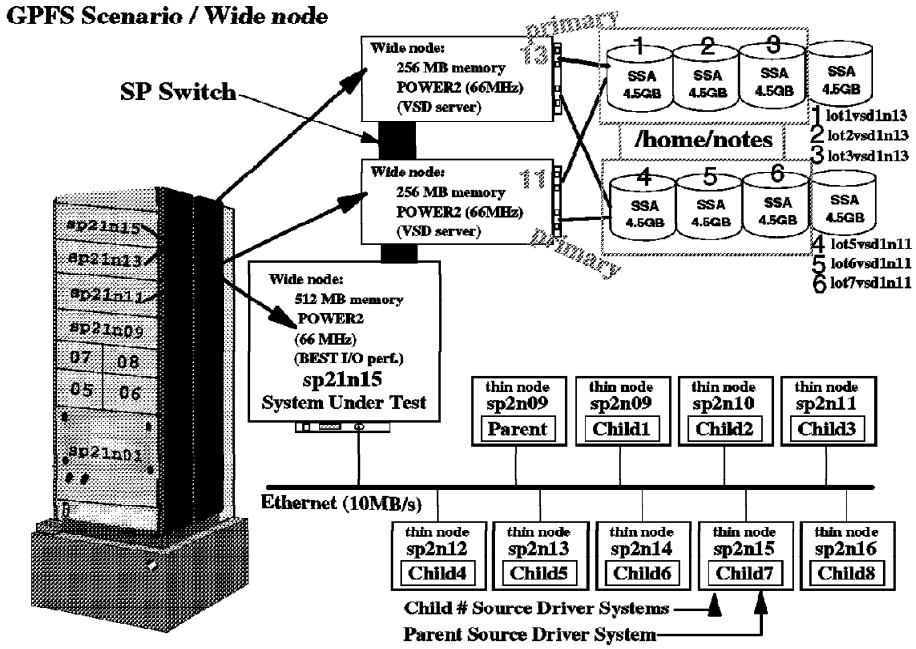


Figure 17. GPFS Wide Node Scenario. Wide RS/6000 SP node with 512MB of RAM and one POWER2 processor as Application server node. Two wide RS/6000 SP nodes with 256MB of RAM and one POWER2 processor as VSD servers. Two loops of four 4.5GB SSA disks shared between the two VSD server nodes.

We used these four different configurations during all benchmarks. The following section describes the procedures we followed to do the tests.

5.1.3.4 Test Methodology

The different configurations of our servers under test were placed under a load of 750 users using the eight drivers described in the previous section connected over an Ethernet LAN.

For the GPFS scenarios, in the servers under test, the GPFS parameters were varied in order to find the best values.

The operating system parameters in all the systems used (the drivers, the servers under test and the VSD servers) were defined at the most appropriate values and were constant for the remaining benchmark sequences.

The only operating system parameter we changed in the servers under test was maxuproc=2000.

As you can see from both the figures and the descriptions of the scenarios, the local JFS I/O configurations were the best ones for Lotus Domino Server.

In all our benchmarks, the server under test was running standard PSSP software; no other applications were running.

In the GPFS scenarios, only the RVSD processes and the GPFS processes were additionally running.

In the local JFS configurations, RVSD and GPFS were stopped.

For all benchmarks, the network traffic was monitored, and in every benchmark, the network was not a bottleneck.

For all benchmarks, the I/O load, CPU load, processes profiles, and so on were monitored.

Each time we ran a new benchmark, we restored the initial *.nsf files we generated by using the following commands:

- `notebnch mailinit`
- `register`

5.1.3.5 Explaining the Data Results

This section provides a summarized description of our tests.

We ran monitoring tools during every test in order to determine the reasons for the behavior of the Lotus Domino Server in each situation. We used the following tools:

- iostat (I/O)
- vmstat (CPU, paging, process queue)
- sar (CPU multiprocessors)
- tprof (processes profiles)

After we finished each test, we ran the command:

```
parent:/home/parent> notesnum maildb 8
```

Refer to Appendix E, “Miscellaneous NotesBench Information” on page 199 for an example of the output that generates this command, and a description of the content.

Table 4 summarizes the results of our tests.

Table 4. Marks of Lotus NotesBench for Lotus Notes R4 for All benchmarks. This table contains the results of all benchmarks.

Scenario	Total Test Time (secs.)	Users	NotesMark: Trans/Min	Response Time (msecs.)
Bench1.High/Local JFS(1)	15180	750	1786	153
Bench2.High/Local JFS(2)	19320	750	1766	152
Bench3.Wide/Local JFS(1)	13980	750	1751	1070
Bench4.Wide/Local JFS(2)	14280	750	1714	2127
Bench5.High/GPFS 64K(1)	16200	750	1754	563
Bench6.High/GPFS 64K(2)	18840	750	1749	695
Bench7.High/GPFS 16K(1)	28140	750	1659	2927
Bench8.High/GPFS 64K(Stripe Group Manager in App. Server)	14760	750	1758	499

1. Data from **local JFS configuration with high node**:

   The results for the local JFS configuration using the high node as Application server node were the best ones for the response time; the NotesMark was the same.

   This is because the Mail and Shared Database test does not deal with heavy I/O usage, and because the local configuration with two balanced loops of SSA disks is the best one you can have. In a later section, you will see the results for the same test, but with using GPFS instead.

   The main considerations when comparing these two different solutions (GPFS versus local) are: with GPFS, two additional layers (RVSD and GPFS) are run; there is a bigger memory requirement; there is more CPU consumption by the mmfsd daemon and the Switch network.

Refer to Appendix E, “Miscellaneous NotesBench Information” on page 199 for some average monitoring outputs.

2. Data from **local JFS configuration with wide node**:

   We make the results for this configuration available so you can analyze the differences between using GPFS or using local JFS for one processor machine.

   The response time with the wide node is slower than with the high node because the NotesBench tool is multithreaded and takes advantage of the SMP architecture in the high node.

   The monitoring data does not contain anything of interest. The wide node was perfectly able to hold the load of the benchmark for 750 users, and the results for this scenario are shown in Table 4 on page 113.

3. Data from **GPFS configuration with high node**:

   This is the best implementation for the Lotus Domino Server if you want to use GPFS as the file server; we found the best values for the GPFS parameters by using this configuration. (Refer to 5.1.1, “GPFS Performance Considerations –” on page 82 for a description of these values.)

   Next, we look at the results of the benchmarks with the following values in the GPFS parameters:

   - Block size 64KB
   - I-node size 1KB
   - Indirect size 8KB
   - maxFilesToCache 2000 (using `lsof`, refer to F.3, “LiSt Open File” on page 213 for more information)
   - `pagepool 80MB`
   - `mallocsize 8MB`
   - prefetchThreads 48
   - worker1Threads 72
   - worker2Threads 24
   - (The rest of the values were the default values.)

   As you can see, we did not follow the rules given in 5.1.1, “GPFS Performance Considerations –” on page 82 regarding pagepool and mallocsize values.

   These rules recommend the following sizes:

```
pagepool = 64KB * 2000 * 2 = 256MB
mallocsize = 4KB * 2000 + 2MB = 10MB
```

In our system under test, we had 512MB of RAM. When we tried to define pagepool and mallocsize by following the rules exactly, the Lotus Domino Server was unable to hold the load of 750 users because it did not have enough memory for Lotus work. We then tried different values for pagepool and mallocsize and found that 80MB for pagepool (data cache) and 8MB for mallocsize (metadata cache) were enough for our environment.

This experience of searching for good values for pagepool and mallocsize led us to the following conclusions:

a. The benchmark does not need too much data cache.

b. However, the size of the metadata cache is really important. (This makes sense because we were dealing with about 2000 small/medium files, with sizes ranging from zero to 2.5MB.)

Further details are given in 5.1.3.6, “Conclusions” on page 116.

Refer to Appendix E, “Miscellaneous NotesBench Information” on page 199 for some average monitoring outputs.

4. Data from **GPFS configuration with wide node**:

After determining the best GPFS parameters to use (as described in 5.1.1, “GPFS Performance Considerations –” on page 82), we tried to run the same benchmarks in the wide node. However, we found that the wide node could not hold the same Lotus Domino Server load when using GPFS. We tried to run the benchmark for 640 users, but in the end it was not able to arrive at a stable state. (We did not try with less than 640 users.)

The main reason for this result was the high CPU consumption by the `mmfsd` daemon when it services data to applications like Lotus Domino Server.

Refer to Appendix E, “Miscellaneous NotesBench Information” on page 199 for some average monitoring outputs.

5. Additional Information

We derived additional information from the third scenario, which ran Lotus Domino Server on the high node using GPFS.

- Stripe Group Manager

  No additional CPU consumption by the `mmfsd` daemon was detected on the VSD server node when forcing the Stripe Group

Manager to be on it. (What is more, we got a slightly better response time).

Also, no additional CPU consumption by the mmfsd daemon was detected on the high node when forcing the Stripe Group Manager to be on it.

- Running the Lotus Domino Server with a block size of 16KB results in lower performance. Obviously you save some disk space, but nowadays disks are less expensive.
- Running the Lotus Domino Server with a block size of 256KB did not improve the performance, and required more pinned memory (pagepool, mallocsize) and disk space.

5.1.3.6 Conclusions

We reached the following conclusions based on the results of the benchmarks.

- The GPFS key parameters you must tune correctly in order to achieve good performance when running Lotus Domino Server using GPFS are:
  - maxFilesToCache
  - pagepool
  - mallocsize
  - prefetchThreads
  - worker1Threads
  - worker2Threads
- The best Application server node to run the Lotus Domino Server using GPFS is a *multiprocessor* node. It is not necessary to have up to six processors, as we had in our benchmarks; you can improve performance with just two processors when running Lotus Domino Server on top of GPFS. As previously mentioned, this is because the GPFS daemon is a multithreaded daemon, and it takes advantage of the multiprocessor nodes.

  You can try to run your Lotus Domino Server using GPFS in either a wide or thin node, but the number of concurrent users supported will decrease and the response time of the server will increase. If you accept these conditions, then you can do it.

  In an Application server node running Lotus Domino Server or similar applications, the CPU load of the mmfsd daemon is high.
- If the I/O configuration for your Lotus Domino Server can be implemented in a standalone machine locally, do not use GPFS; in

this case, by using a local I/O configuration, you will get better performance.

- GPFS is highly scalable. It allows you to spread the entire I/O load over the VSD server nodes of the GPFS cluster, thus removing the I/O capacity limitations on the nodes.
- You must balance, as much as possible, the I/O load among the VSD server nodes of the GPFS cluster. Do the same with the disks inside each VSD server node.
- You can reduce the memory requirements for GPFS. If possible, follow the rules explained in 5.1.1.1, “Performance Parameters” on page 83. You can also reduce the values for pagepool and mallocsize, but as we have seen, this can impact the performance of your Lotus Domino Server.

  Try to find out from your application how much data cache it will need. Metadata cache (mallocsize) is directly proportional to the number of files your application will deal with.
- You can place the Stripe Group Manager in the Application server node or in VSD server nodes. In our benchmark, we did not detect any additional CPU load from the Stripe Group Manager. However, depending on the number of GPFS file systems that GPFS is managing and the work they are giving to it, you can be forced to move the Stripe Group Manager to another node of the GPFS cluster.
- There is high flexibility in managing the GPFS file system resources. After running the benchmarks, we tested the management facilities. We found that you can remove or add a disk to the Lotus Domino Server GPFS file system while the Lotus Domino Server is running.
- Using GPFS replicas provides real benefits. Replicating data and metadata provides both high availability and performance improvement for Lotus installations with a heavy read workload (performing full-text searches, for example).

5.1.4 Moving the Lotus Domino Server between RS/6000 SP Nodes

It is difficult to perform maintenance tasks on a production system that is already up and running and providing certain services to your users. However, if the system is running a Lotus Domino Server, you should be able to maintain the service available for your users, as explained in this section.

Notes

1. In a GPFS environment, you will benefit from additional flexibility when moving your Lotus Domino Server between nodes of your GPFS cluster.
2. In this section, we use the term *primary node* for the GPFS node that is going out of service for maintenance reasons, and *backup node* for the GPFS node that will impersonate the primary.
3. Only TCP/IP procedures are explained.

1. Hardware requirements:
   - One additional node must be available to run the Lotus Domino Server in the GPFS cluster; this will be the backup node.
   - One additional network adapter must be ready to use in the backup node.
2. Software installation and configuration requirements:
   - Lotus Domino Server binaries must be locally installed in the backup node.
   - The additional network adapter must be configured to allow your Lotus Notes clients to access the Lotus Domino Server transparently.
   - Depending on how you plan to connect and configure the network adapter of the backup node, you will need additional software configuration.
3. User environment considerations:

   If you are not using the AIX automounter and File Collections, you must replicate the same user configuration you have in the primary node for the user notes in the backup node.

   AIX automounter

   As you know, when a user logs on to a node that is not located in his home directory, the AIX automounter transparently mounts the remote directory (of the remote node you defined when creating the user) in the local node. Therefore, if you are moving your Lotus Domino Server when you log into the backup node as user notes, the AIX automounter will do this.

   To avoid using NFS when running the Lotus Domino Server in the backup node, you only have to change the actual directory from /u/notes to /home/notes before you start the Lotus Domino Server.

Following are two different ways to move your Lotus Domino Server from one GPFS node to another in the same cluster:

- **You can use exactly the same TCP/IP configuration of the Lotus Domino Server network adapter of the primary node.**

  Follow these steps:

  1. Mount the GPFS file systems containing the Domino Server databases in the backup node (if it is not already mounted).
  2. Stop the Lotus Domino Server in the primary node.
  3. Set down the Lotus Domino Server network interface of the primary node.
  4. Set up the same TCP/IP configuration of the Lotus Domino Server network adapter of the primary node in the additional network adapter of the backup node.
  5. Start TCP/IP in the Lotus Domino Server network interface of the backup node.
  6. Log on to the backup node as user notes. (If using AIX automounter, execute: cd /home/notes.)
  7. Start the Lotus Domino Server in the backup node.

     The service is already up.

  Example:

  - The notes user is didac.
  - The home directory is /home/didac in the primary node sp21n15.
  - The Domino Server databases directory is /home/didac/notesr4.
  - The TCP/IP network interface in the primary node is tr1.
  - The backup node is sp21n01.
  - The TCP/IP network interface in the backup node is tr0.

  1. Log on to backup node sp21n01 as user root and type in:

     ```
     sp21n01:/> mount /home/didac
     ```

  2. On the primary node sp21n15, stop your Lotus Domino Server in your preferred way. For example, if you have the Lotus Domino Server running in a dedicated window, enter the quit command in it, as shown:

```
Lotus Domino  Server (Release 4.5a (Intl) for UNIX) 22/09/97 11:09:12

Server name:            didac/IBMITSO
Server directory:       /home/didac/notesr4
Elapsed time:           00:01:33
Transactions/minute:    Last minute: 0; Last hour: 0; Peak: 0
Peak # of sessions:     0 at
Transactions:           0
Shared mail:            Not enabled
Pending mail:  0        Dead mail:  0
>
22/09/97 11:09:49     0 Transactions/Minute, 0 Users
>QUIT
```

3. On the primary node, as user root, type in:

```
sp21n15:/> ifconfig tr1 down
```

4. Once the network interface on the primary node is down, set up the backup interface. If you already have the backup network interface defined with the same parameters as the primary one, you only need to bring up the backup network interface as follows:

   Log on to the backup node as user root and type in:

```
sp21n01:/> ifconfig tr0 up
```

   However, if the network interface of the backup node is not already configured, do the following.

   Log on to the backup node as user root and type in:

```
sp21n01:/> smitty chinet
```

   This screen will be displayed:

```
                     Available Network Interfaces

 Move cursor to desired item and press Enter.

   en0     Standard Ethernet Network Interface
   et0     IEEE 802.3 Ethernet Network Interface
   TR0    TOKEN RING NETWORK INTERFACE

 F1=Help                 F2=Refresh              F3=Cancel
 F8=Image                F10=Exit                Enter=Do
 /=Find                  n=Find Next
```

   After you select the **tr0** interface, enter the same parameters you have in the primary network adapter in the Change / Show a Token-Ring Network Interface screen:

```
                Change / Show a Token-Ring Network Interface

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
  Network Interface Name                              tr0
  INTERNET ADDRESS (dotted decimal)                  [9.10.100.3]
  Network MASK (hexadecimal or dotted decimal)       [255.255.255.0]
  Current STATE                                       up
  Use Address Resolution Protocol (ARP)?              yes
  Enable Hardware LOOPBACK Mode?                      no
  BROADCAST ADDRESS (dotted decimal)                 []
  Confine BROADCAST to LOCAL Token-Ring?              no



F1=Help             F2=Refresh          F3=Cancel           F4=List
Esc+5=Reset         F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

5. After the TCP/IP network interface is up on the backup node, log on to the backup node as user didac and type in:

```
sp21n01:/home/didac> server
```

   After the complete start of the Lotus Domino Server, the Lotus Notes Service is already up.

The second method of moving your Lotus Domino Server from one GPFS node to another in the same cluster is as follows:

- **You can use a different IP address in the Lotus Domino Server network adapter of the backup node.**

Notes

1. You must set the *same hostname* in the Lotus Domino Server network adapter of the backup node as you used in the Lotus Domino Server network adapter of the primary node. You can change the IP address, but you *cannot* change the hostname.
2. This procedure can be useful if you need to keep the Lotus Domino Server TCP/IP network interface up and running in the primary node during maintenance work.
3. You can use this procedure only if you have full control of the name and IP address resolution of your Lotus Notes clients.
4. The Lotus Notes clients cannot use local name resolution. They should only be configured to use Domain Name Services, and you must guarantee each change in the DNS servers will be noticed in all the Lotus Notes clients.

   (In fact, you can perform this procedure, with Lotus Notes clients resolving names and IP addresses in local, but is your responsibility to change the local configuration of all your Lotus Notes clients, which can be difficult and complicated, depending on your installation.)
5. If you plan to connect the Lotus Domino Server network adapter of the backup node in a different subnet of your organization, you must be completely sure that your Lotus Notes clients are able to physically and logically access this subnet (*physical path* and *routing path*).

Follow the next steps:

1. Mount the GPFS file systems containing the Domino Server databases in the backup node (if it is not already mounted).
2. Stop the Lotus Domino Server in the primary node.
3. Set up the TCP/IP of the Lotus Domino Server network adapter in the backup node.
4. Start TCP/IP in the Lotus Domino Server network interface of the backup node.
5. Change your DNS configuration and REFRESH the named daemon.
6. Check the DNS changes. Be sure all the Lotus Notes clients get the new IP address when asking for the name of the Lotus Domino Server.

7. Log on to the backup node as user notes. (If using AIX automounter, execute cd /home/notes.)
8. Start the Lotus Domino Server in the backup node. The service is already up.

Example:

- Notes user is didac.
- The home directory is /home/didac in the primary node sp21n15.
- The Domino Server databases directory is /home/didac/notesr4.
- The TCP/IP network interface in the primary node is tr1.
- The IP address of the Lotus Domino Server in the primary node is 9.10.100.3.
- The backup node is sp21n01.
- The TCP/IP network interface in the backup node is tr0.
- The IP address of the Lotus Domino Server in the backup node is 9.10.100.24.
- The actual DNS configuration is as follows.

  The file named.data:

```
sp21tr01         99999999 IN     A       9.10.100.24
sp21tr15         99999999 IN     A       9.10.100.3
# Lotus Notes Aliases
DIDAC           99999999 IN     CNAME   SP21TR15
```

  The file named.rev:

```
3          IN PTR sp21tr15.msc.itso.ibm.com.
24         IN PTR sp21tr01.msc.itso.ibm.com.
```

  As you see, we used aliases for the name of the Lotus Domino Server. We recommend you do the same to make your work easier.

In the following section, we describe only the DNS change steps, since the remaining steps are exactly the same as in the previous example.

To change the DNS configuration:

1. Log on to the DNS server machine as user root.
2. Using your favorite editor, make the following change in the named.data file:

```
sp21tr01         99999999 IN     A       9.10.100.24
sp21tr15         99999999 IN     A       9.10.100.3
# Lotus Notes Aliases
DIDAC           99999999 IN     CNAME   SP21TR01
```

3. Now you must update the DNS daemon information. Run the command:

```
riscserver:/> refresh -s named
```

4. Verify the DNS change by checking in one of the Lotus Notes clients, as follows:

```
notesclient:/> host didac
sp21tr01.msc.itso.ibm.com is 9.10.100.24, Aliases: didac.msc.itso.ibm.com
```

Alternatively, you can just ping it:

```
notesclient:/> ping didac
PING sp21tr01.msc.itso.ibm.com: (9.10.100.24): 56 data bytes
64 bytes from 9.10.100.24: icmp_seq=0 ttl=254 time=2 ms
64 bytes from 9.10.100.24: icmp_seq=1 ttl=254 time=2 ms
64 bytes from 9.10.100.24: icmp_seq=2 ttl=254 time=2 ms
64 bytes from 9.10.100.24: icmp_seq=3 ttl=254 time=2 ms
```

5.1.5 Migrating Lotus Domino Server to GPFS

Attention

There is no way to *directly migrate* from AIX JFS file systems to GPFS file systems. Instead, you have to create the new GPFS file systems and copy all the data from the AIX JFS file systems to the new GPFS file systems.

5.1.5.1 Migrating from a RS/6000 - AIX Standalone Installation

1. Migration of the user environment:

   Follow the process explained in section 5.1.2.3, “Create and Configure the User Environment” on page 97 to set up the user environment in the RS/6000 SP system for Lotus Notes users.

2. Migration of the file systems:

   Install the Domino Server binaries in a local JFS file system in the GPFS node selected for the migration.

   Create the new GPFS file system for the Domino Server databases.

   Back up all your JFS file systems where you have the Domino Server databases, and restore them in the new GPFS file system, preserving owner, group and permissions.

5.1.5.2 Migrating from a RS/6000 SP Node Standalone Installation

1. Migration of the file systems:

   Install the Domino Server binaries in a local JFS file system in the GPFS node selected for the migration.

   Create the new GPFS file system for the Domino Server databases.

   Back up all your JFS file systems where you have the Domino Server databases, and restore them in the new GPFS file system, preserving owner, group and permissions.

5.1.6 When to Run Lotus Domino Server over GPFS

The use of Lotus Domino Server over GPFS is recommended in the following cases:

- When very large I/O capacity is required (that is, when you require scalability).

  If you need to implement a very large Lotus Notes database system, and the I/O capacity of a standalone machine is insufficient (for example, in big document databases with a high read load from clients), GPFS provides a scalable solution for your Lotus Domino Server

- When you want easy and flexible file system management.

  GPFS allows easy and flexible management of the GPFS file system resources.

  - You can add/delete disks even while file systems are being mounted and applications are being run (and you can combine this task with replication, if necessary).
  - You can add/delete nodes from the cluster.
  - You can rebalance a file system across the actual disks.
  - You can run administration tasks from any node of the GPFS cluster.
  - You can move the Lotus Domino Server among GPFS nodes easily.

  If these capabilities are important for your installation, GPFS provides an effective solution.

- When you need high availability.

  Because using GPFS allows you to have two data replicas and two metadata replicas, you can achieve high availability in massive read environments.

- Future Improvements.

  As shown in Figure 12 on page 82, if Lotus Domino Server modifies its actual locking behavior (as, say, in a Lotus Domino Server Parallel Edition), then using GPFS will be the most efficient way to implement clusters of Lotus Domino Server. In this way, you can avoid the CPU usage and network traffic of any replicas, as well as other cluster overhead.

5.2 MPI Applications

In most cases, a user application that runs on the SP can use GPFS in place of PIOFS or NFS without altering the application. GPFS combines the shared access of NFS with the high performance of PIOFS. Refer to Chapter 4, “Migration” on page 75 for more details on migration to GPFS.

If an application spends significant amounts of time doing I/O, it is probably worth investing effort to see if different approaches to I/O can achieve better performance. Two synthetic applications are presented in this section. The first stresses the metadata part of a file system. The second exercises the read/write characteristics of a file system to see how block size, access patterns and other IO parameters affect performance.

5.2.1 Synthetic Applications

The following two applications were tested on a small SP system. The system consisted of SP2 thin nodes, an HPS, internal SCSI and 9333 disk units. All of these components are slower than the currently available components (P2SC and SMP nodes, SP switch and SSA drives). The applications are not meant to demonstrate the best possible performance, but rather the relative performance of various I/O techniques.

The following file systems were tested:

- /dev/null (for write tests)
- JFS on the internal SCSI disks with logs on a different spindle
- GPFS via the HPS on one server using type internal SCSI disk
- GPFS via the HPS on two servers using type internal SCSI disks
- GPFS via the HPS on one server using type 9333 disk

The tests were run on one, two, and four nodes at a time. Source code for the synthetic applications is available. Refer to Appendix F, “How to Get the Examples in This Book” on page 213 for more information.

There are many configuration options for GPFS. The file systems for these tests used the installation defaults. Many of the tuning trade-offs are discussed in Chapter 3, “Failover Scenarios” on page 57.

5.2.1.1 Metadata Application

Note: This command is called `pmd`, but it is in no way related to the process in the Parallel Environment (POE) which has the same name.

The metadata application will create a number of files, read through the directory stat()ing the files, and then remove the files. In addition to these tasks, a background thread can consume the remaining processor time. The application tells how much time the average `open()`, `readdir()`, `stat()` and `unlink()` calls take. The amount of processor time left for other tasks can be determined by the background thread.

If more than one node is used to run the test, a verification of the `O_CREAT` combined with `O_EXCL` is performed. There is also an option to force the times at which the metadata operations occur to be synchronized on different nodes by use of the MPI_Barrier() routine.

The test has two main parts. First, all active nodes create, read/stat and remove their files in a common directory. Second, each node does the same creates, read/stats and removes, but the files are stored in a new, node-specific directory. This allows us to see if contention in the same directory makes a difference in performance.

This test is similar to some of the operations that would in occur in a task such as a `make` on a collection of files that generate small objects or that is mostly up-to-date. This sort of task would generate many operations on the metadata parts of the file system.

Figure 18 on page 128 and Figure 19 on page 129 show sample reports from the metadata application:

```
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>: ./pmd -procs 1 100 N N /tmp pgc

**** Ran from 1997/10/02 16:07:20 to 1997/10/02 16:07:28 on 1 nodes. ****
./pmd 100 N N /tmp pgc
 Calibration Overhead (removed from stats)
 AVE=1.020067e-05, STDEV=6.238373e-06, MIN=1.000002e-05, MAX=5.799750e-04

The semantics on open() do work as expected

Different file names in same directory:

 Barrier
 AVE=4.324958e-07, STDEV=7.065855e-07, MIN=-1.006804e-07, MAX=4.299324e-06
 Create
 AVE=1.247060e-02, STDEV=3.762534e-03, MIN=7.628649e-03, MAX=2.293027e-02
 Stat
 AVE=1.547759e-03, STDEV=2.809541e-04, MIN=7.537434e-05, MAX=3.515974e-03
 Unlink
 AVE=1.256536e-02, STDEV=3.700915e-03, MIN=9.183749e-03, MAX=2.241187e-02

Same file names in different directory:
 Barrier
 AVE=4.506641e-07, STDEV=7.325762e-07, MIN=-1.006804e-07, MAX=4.224323e-06
 Create
 AVE=1.280494e-02, STDEV=4.435264e-03, MIN=9.383749e-03, MAX=3.384115e-02
 Stat
 AVE=1.642334e-03, STDEV=7.267294e-04, MIN=2.012743e-04, MAX=7.824674e-03
 Unlink
 AVE=1.259110e-02, STDEV=3.830179e-03, MIN=9.144199e-03, MAX=2.315257e-02

wall time =   5.4229,  user time =   0.0500, sys time =   0.2700
wall time - (user + system time) =   5.1029 (other system time)
This process got  5.901% of the available CPU time
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>: ./pmd -procs 2 100 N N /gpfs52 pgc

**** Ran from 1997/10/02 13:29:37 to 1997/10/02 13:30:49 on 2 nodes. ****
./pmd 100 N N /gpfs52 pgc
 Calibration Overhead (removed from stats)
 AVE=1.013996e-05, STDEV=1.149876e-06, MIN=1.004999e-05, MAX=4.867499e-05

The semantics on open() do work as expected

Different file names in same directory:

 Barrier
 AVE=2.272037e-06, STDEV=3.117955e-06, MIN=-3.997288e-08, MAX=1.341004e-06
 Create
 AVE=1.714458e-01, STDEV=1.176691e-01, MIN=1.340461e-02, MAX=5.157580e-01
 Stat
 AVE=1.869917e-03, STDEV=9.807880e-04, MIN=1.527535e-03, MAX=7.946610e-03
 Unlink
 AVE=2.290273e-01, STDEV=7.354260e-02, MIN=9.951071e-02, MAX=3.666961e-01
```

Figure 18. Example Output from the Metadata Application (Part I)

```
Same file names in different directory:

 Barrier
 AVE=2.255871e-06, STDEV=3.043095e-06, MIN=-3.997288e-08, MAX=8.335033e-06
 Create
 AVE=2.635205e-02, STDEV=2.503719e-02, MIN=9.509910e-03, MAX=1.264067e-01
 Stat
 AVE=1.537612e-03, STDEV=1.830127e-04, MIN=7.548505e-05, MAX=2.541310e-03
 Unlink
 AVE=1.340832e-01, STDEV=2.113726e-02, MIN=1.000587e-01, MAX=3.000125e-01

wall time =  59.6852,  user time =   0.0700, sys time =   0.3500
wall time - (user + system time) =  59.2652 (other system time)
This process got  0.704% of the available CPU time
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>: ./pmd -procs 2 100 N T /gpfs52 pgc

**** Ran from 1997/10/02 14:42:24 to 1997/10/02 14:43:58 on 2 nodes. ****
./pmd 100 N T /gpfs52 pgc
 Calibration Overhead (removed from stats)
 AVE=1.024692e-05, STDEV=1.003669e-05, MIN=1.002499e-05, MAX=9.890750e-04

The semantics on open() do work as expected

Different file names in same directory:

 Barrier
 AVE=1.813075e-06, STDEV=2.635863e-06, MIN=-1.469354e-07, MAX=6.803088e-06
 Create
 AVE=1.567091e-01, STDEV=1.071638e-01, MIN=1.220050e-02, MAX=5.199287e-01
 Stat
 AVE=1.703296e-03, STDEV=4.296528e-04, MIN=1.533203e-03, MAX=4.022678e-03
 Unlink
 AVE=2.089606e-01, STDEV=6.224795e-02, MIN=9.995893e-02, MAX=4.109136e-01

Same file names in different directory:

 Barrier
 AVE=2.381076e-06, STDEV=9.157463e-06, MIN=-1.469354e-07, MAX=1.460531e-06
 Create
 AVE=2.733674e-02, STDEV=2.859700e-02, MIN=9.477103e-03, MAX=1.628778e-01
 Stat
 AVE=2.232533e-03, STDEV=1.656475e-03, MIN=7.502808e-05, MAX=1.103763e-02
 Unlink
 AVE=1.577595e-01, STDEV=5.201136e-02, MIN=1.009233e-01, MAX=4.905361e-01

wall time =  60.7456,  user time =  51.6200, sys time =   1.2800
wall time - (user + system time) =   7.8456 (other system time)
This process got 87.085% of the available CPU time
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>:
```

Figure 19. Example Output from the Metadata Application (Part II)

The cost of the overhead of the loops and measurement code is determined when the test starts. The code in Figure 20 on page 130 shows how this is calculated. The times returned by gettime() is in units of seconds in the form of a double. This amount of time is subtracted from individual samples.

```
/*****************************************************************/
/*
 * this could be a subroutine if needed
 */

#define gettime()         MPI_Wtime()


/*****************************************************************/

void
doCalibrateTest(struct statsStuff_t *statP, long N) {
double t;
long i;

  for (i = 0 ; i < N ; i++ ) {
    t = gettime();
    t = gettime() - t;
    insertSample(statP, t);
  }
}

/*****************************************************************/
```

Figure 20. Function to Determine Measurement Overhead

Next, a message indicates whether the semantics of the open system call using the O_EXCL flag are the same as for a local JFS file system. The routine in Figure 21 on page 131 is used to determine if the semantics work as expected.

```
/*****************************************************************/

void
semanticsTest() {
char buffer·MAXPATHLEN‘;
long i;
long successes;
int rc;
int rcSummed;

  successes = 0;

  for (i = 0 ; i < loops ; i++) {
    sprintf(buffer, "%s/%s%ld", where, prefix, i);
    MPI_Barrier(MPI_COMM_WORLD);
    rc = open(buffer, O_CREAT | O_EXCL | O_RDWR, S_IRUSR | S_IWUSR );
    if (rc < 0) {
      rc = 0;   /* failed */
    } else {
      close(rc);
      unlink(buffer);
      rc = 1;   /* succeeded */
    }

    MPI_Barrier(MPI_COMM_WORLD); /* this should not be needed but... */
    rcSummed = -1;
    MPI_Reduce(&rc, &rcSummed, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
    if (rcSummed == 1) {
      successes++;     /* this will not be used on any task but task 0 */
    }
  }

  if (successes == loops) {
    semanticsWork = 1;
  } else {
    semanticsWork = 0;
  }
}

/*****************************************************************/
```

Figure 21. Function to Verify Semantics of O_EXCL on open()

The report next shows the statistics for performing the operations in a single shared directory, and then statistics for per node directories reported. You can see that the same tasks are done in both cases in Figure 22 on page 132. However, while the same tests are run in both cases, the base names are changed. In the second case, directories must be created and removed.

```
/*****************************************************************/

void
doTest(double overhead) {
char buffer·MAXPATHLEN‘;
int rc;

  /* do the different filename/same directory case */
  sprintf(buffer, "%s/%s%d_", where, prefix, thisTask);
  doCreateTest(&createStats1, &barrierStats1, loops, overhead, buffer);
  doReadStatTest(&statStats1, &barrierStats1, loops, overhead, where);
  doUnlinkTest(&unlinkStats1, &barrierStats1, loops, overhead, buffer);

  /* then the same filename/different directory case */
  sprintf(buffer, "%s/%s%d/", where, prefix, thisTask);
  rc = mkdir(buffer, S_IRUSR | S_IWUSR | S_IXUSR);
  if (rc < 0) {
    fprintf(stderr, "problem with mkdir(%s), errno = %d\n", buffer, errno);
    exit(2);
  }

  MPI_Barrier(MPI_COMM_WORLD);
  doCreateTest(&createStats2, &barrierStats2, loops, overhead, buffer);
  doReadStatTest(&statStats2, &barrierStats2, loops, overhead, buffer);
  doUnlinkTest(&unlinkStats2, &barrierStats2, loops, overhead, buffer);


  rc = rmdir(buffer);
  if (rc < 0) {
    fprintf(stderr, "problem with rmdir(%s), errno = %d\n", buffer, errno);
    exit(2);
  }
}

/*****************************************************************/
```

Figure 22. Function That Tests the Two Primary Cases of Contention

The output ends with information about the amount of wall time, user space time and system time used by the process. Notice that these last three statistics are only for the time spent performing the two main tests. The time needed to initialize the processes, verify the O_EXCL flag, calculate the statistics and print the results are not counted.

The calibrating subtraction may cause some MIN time values can be negative in value. Also, depending on the command line arguments, Barrier will report random values if the barrier functions are not used.

Note the following details from the examples:

- In the first example, shown in Figure 18 on page 128, in the command line `./pmd -procs 1 100 N N /tmp pgc`, the `-procs 1` causes

only a single parallel process to run. The first `N.` on the line indicates we do not want to issue `MPI_Barrier()` calls before the operations. To issue the barrier calls, we would use a `B`. The second `N` indicates we do not want to run the extra thread that will consume all remaining CPU time on the nodes. If we wanted to run the extra thread, we would replace that `N` with a `T`.

100 files will be created in /tmp, a JFS file system, using the string `pgc` to help generate the file names.

- In the second example, each of two nodes is creating 100 files. This time two parallel processes will be used, no barrier calls are made, and the extra thread will not execute. The files will be created in `/gpfs52`, a GPFS file system using a single VSD server in this case, and the string `pgc` is again used to create unique names.
- The third example causes the extra thread to execute, so we can see how much processor time is used by the GPFS and other daemons on the nodes.

  In the second example, the process used 0.070 + 0.350 = 0.420 seconds of the 59.685 seconds that it took the test to run. Less than 1% of the time was used by the process; the rest went to other processes and idle or wait time.

  In the third example, the process used 51.620 + 1.280 = 52.900 seconds of the 60.746 seconds. Since the extra thread is a never-exiting loop, we know any CPU time not used by our process was consumed by other processes on the node. In this case, about 13% of the time was used by these other processes.
- In the JFS case with the extra thread running (not shown), we find the process gets almost 97% of the CPU. This indicates the base level of daemon activity on the nodes is about 3% of the CPU resource. From this we can see that, while doing this kind of work, the GPFS daemons consume about 10% of the CPU resources on the node. This is just one example of GPFS usage. Variations from 5 to 25% have been observed.

  Keep in mind that on faster nodes the GPFS daemons should use relatively less CPU.

Now that we have seen what the report describes, let us examine other results from the metadata application.

5.2.1.2 Metadata Application Results

We first consider the performance of GPFS relative to JFS for the single node case. In this test, two local JFS file systems are compared to a GPFS file system that is housed on a remote VSD. One JFS file system

shares a single disk with its log while the second JFS uses an different disk. The disk hardware is the same for the JFS and VSD.

Table 5. GPFS Vs. JFS Metadata Performance Single Node Case (mSec)

Operation	JFS	JFS Diff Log Disk	GPFS
Files in Common Directory			
Create	13.0	12.7	18.6
Readdir/Stat	1.7	1.7	1.6
Unlink	12.6	12.5	91.4
Files in Different Directory			
Create	13.6	12.7	16.7
Readdir/Stat	1.7	1.7	1.6
Unlink	12.5	12.5	98.1
Wall Time (sec)	5.6	5.6	29.0

As shown in Table 5, in most cases, with similar disk hardware, the performance is comparable. In the `readdir()/stat()`, GPFS is faster than JFS. The GPFS `unlink()` case is a good deal slower than the JFS case. As expected, JFS is a bit faster when the JFS log is placed on a different disk.

In most tasks, `readdir()/stat()` operations are done more often than files are created and deleted. This minimizes the impact of the `create()/unlink()` calls.

Table 6 and Table 7 on page 135 show how the performance scales when multiple clients use GPFS at the same time.

Table 6 (Page 1 of 2). GPFS Performance with Multiple Clients (no Barrier) (mSec)

Operation	1 Node	2 Nodes	4 Nodes
Files in Common Directory			
Create	18.6	180.0	432.0
Readdir/Stat	1.6	1.7	1.7
Unlink	91.4	172.0	609.0
Files in Different Directory			
Create	16.7	26.9	36.9
Readdir/Stat	1.6	1.7	1.8

Table 6 (Page 2 of 2). GPFS Performance with Multiple Clients (no Barrier) (mSec)			
Operation	**1 Node**	**2 Nodes**	**4 Nodes**
Unlink	98.1	161.0	345.0
Wall Time (sec)	29.0	68.2	159.4

Table 7. GPFS Performance with Multiple Clients (Barrier) (mSec)			
Operation	**1 Node**	**2 Nodes**	**4 Nodes**
Files in Common Directory			
Create	15.6	169.0	353.0
Readdir/Stat	1.6	1.7	3.6
Unlink	89.7	180.0	429.0
Files in Different Directory			
Create	18.9	26.2	46.7
Readdir/Stat	1.5	1.7	2.5
Unlink	94.0	159.0	314.0
Wall Time (sec)	28.6	73.2	162.6

In most cases, the performance of `readdir()/stat()` continues to be good when more clients are active. This degrades slightly when barriers are used and the nodes try to do this operation at about the same instant. When the nodes are not synchronized, they appear to fall into a pattern that allows better performance.

Note the big increase in the time needed to create a file which occurs when going from single to multiple node, and all files are using a common directory. After the big jump when going from one to two nodes, the performance falls off relatively less when moving from the two-node to the four-node case.

This does not happen to such a degree when the nodes use independent directories to reduce contention. Notice how the create operations take about one-tenth the time when the nodes do not have to share a directory with other nodes. The `unlink()` operations do not degrade to the same degree, but `unlink()` started with slower operation.

5.2.1.3 Metadata Application Summary

When we look at wall time, we see that as client nodes are added, the time needed to do metadata operations increases. If your parallel process must create and delete large numbers of files, you will be better

off having a single node perform all of these operations. On single machine applications, GPFS performs quite well. For the very common `readdir()/stat()` operations, GPFS outperforms JFS.

Unless your parallel application does an extensive number of file creates or deletes, you are reasonably safe to ignore changing your code for GPFS.

5.2.1.4 Read and Write Application

The second application will `write()` or `read()` the number of blocks that are specified on the command line. The size of the I/O operations, as well as the stride of the operations, can be varied. The file position offset, memory buffer offset, and the file access pattern can be altered. As with the metadata application, a thread can be run in the background to indicate how much processor time is consumed by the file system code, and `MPI_Barrier()` can be used to force synchronization between nodes.

```
pgc@sp2en0 </u/pgc/ITSO>: ./pio
wrong number of parameters
usage:  ./pio N iosize buf_align file_align R|W B|N file
        N is number of loops of read()/write()
        iosize is size of buffer [0-524288]
        buf_align is buffer offset [0-4095]
        file_align is file io offset [0-(iosize-1)]
        R|W is Read for Write file
        B|N is Barrier on io times or No barrier
        C|S do IO in chunk or stripe pattern
        S|N is use O_SYNC on open()
        T|N use a background thread to consume CPU
        file is name of file to read()/write()
pgc@sp2en0 </u/pgc/ITSO>:
```

Figure 23. Example Output from the I/O Application (Part 1 of 2)

```
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>: ./pio 8192 4096 0 0 W N S N N /dev/null

**** Ran from 1997/10/06 11:34:30 to 1997/10/06 11:34:31 on 1 nodes. ****
./pio 8192 4096 0 0 W N S N N /dev/null
 Calibration Overhead (removed from stats)
 AVE=5.303538e-06, STDEV=8.214410e-07, MIN=5.224996e-06, MAX=5.982500e-05

 Barrier
 AVE=5.640640e-06, STDEV=8.519675e-07, MIN=5.324997e-06, MAX=3.980000e-05
 IO
 AVE=2.236255e-05, STDEV=8.414528e-06, MIN=2.172499e-05, MAX=6.138500e-04

Wall Time = 000.3352  User Time = 000.2500  System Time = 000.0800
wall time - (user + system time) =   0.0052 (other system time)
This process got 98.436% of the available CPU time
Total data rate was 100.0900Mbytes / second
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>: ./pio 8192 524288 0 0 W N S N N /dev/nul

**** Ran from 1997/10/06 11:36:05 to 1997/10/06 11:36:05 on 1 nodes. ****
./pio 8192 524288 0 0 W N S N N /dev/null
 Calibration Overhead (removed from stats)
 AVE=5.351700e-06, STDEV=5.728213e-06, MIN=5.224996e-06, MAX=5.757250e-04

 Barrier
 AVE=5.692780e-06, STDEV=4.328667e-06, MIN=5.474998e-06, MAX=3.643250e-04
 IO
 AVE=2.229261e-05, STDEV=5.120971e-06, MIN=2.172499e-05, MAX=4.595750e-04

Wall Time = 000.3349  User Time = 000.1700  System Time = 000.1700
wall time - (user + system time) =  -0.0051 (other system time)
This process got 101.512% of the available CPU time
Total data rate was 12823.2831Mbytes / second
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>:
pgc@sp2en0 </u/pgc/ITSO>: ./pio 8192 524288 0 0 W N C N N /dev/nul

**** Ran from 1997/10/06 11:39:00 to 1997/10/06 11:39:00 on 1 nodes. ****
./pio 8192 524288 0 0 W N C N N /dev/null
 Calibration Overhead (removed from stats)
 AVE=5.368165e-06, STDEV=5.630581e-06, MIN=5.224996e-06, MAX=5.591000e-04

 Barrier
 AVE=5.421338e-06, STDEV=4.427061e-06, MIN=5.324997e-06, MAX=3.990500e-04
 IO
 AVE=2.345661e-05, STDEV=7.779135e-05, MIN=2.172501e-05, MAX=6.077775e-03

Wall Time = 000.2927  User Time = 000.2300  System Time = 000.0500
wall time - (user + system time) =   0.0127 (other system time)
This process got 95.655% of the available CPU time
Total data rate was 14672.6857Mbytes / second
pgc@sp2en0 </u/pgc/ITSO>:
```

Figure 24. Example Output from the I/O Application (Part 2 of 2)

Figure 23 on page 136 and Figure 24 show example runs of the `pio` application. This example simply displays usage information of the command. In the first example, shown in Figure 24, the `pio` application will perform 8192 I/O operations on a buffer of size 4096. The buffer will be aligned on the VM page and the file position will be aligned on page boundaries (the 0 0 following 4096). The I/O operations will be writes because of the `W`. The `W` is replaced by `R` to perform reads.

In this case, we do not use barriers to synchronize the operation on multiple nodes (there is only one), and we do not use the extra thread or the `O_SYNC`. The buffers are written to the file /dev/null.

The `C|S` parameter is set to `S`. This causes `pio` to perform an `lseek()` between each I/O, so the buffers are written out in stripes rather than in a continuous chunk, as would be done if `C` was used.

The second example, also shown in Figure 24 on page 137, is the same as the first except that the size of the buffer is 524288 bytes instead of 4096. We can see that the time to do an I/O is about the same in both cases (2.236e-5 versus 2.229e-5), so the I/O rates are determined by the number of bytes transferred in each I/O operation.

In the last example in Figure 24 on page 137, the data is written as a continuous set of blocks rather than as a series of stripes. Since there is no need to call *lseek()* between the *write()* functions, the I/O rate is even faster. This last test of writes to /dev/null shows how fast AIX can move data in the best possible conditions.

The report lists the time the command ran, followed by the parameters that were used. As with the `pmd` application, the overhead is given. The statistics on the barrier and I/O times are given in seconds. Again, as with *pmd*, the overall test times are given. This application adds a line to indicate the total data rate, which is calculated by dividing the total number of bytes by the wall time.

```
buffer = malloc(iosize + ( 2 * PAGESIZE ) );  /* need some space for all*/
if (buffer == NULL) {
  fprintf(stderr, "can't malloc() buffer\n");
  exit(3);
}
obuffer = (char *) (bufferAlign +
                    ((((long) buffer) / PAGESIZE ) * PAGESIZE)
                   );

if (cS == 'C') {
  rc = lseek(fileD, ((thisTask * iosize * loops) + fileAlign), SEEK_CUR
} else {
  rc = lseek(fileD, ((thisTask * iosize) + fileAlign), SEEK_CUR);
}
if ( rc == -1 ) {
  fprintf(stderr, "can not lseek(), errno = %d\n", errno);
  exit(3);
}

for ( i = 0 ; i < iosize ; obuffer·i++‘ = (0x80 + thisTask) ) ; /* page */
```

Figure 25. Setup for IO Application

The code that sets up the test loop is shown in Figure 25. First a buffer with the desired alignment is created, and then the initial lseek() is performed on the file. Finally, all bytes in the buffer are touched to make sure everything is in real memory.

```
for (i = 0 ; i < loops ; i++ ) {
  tBarrier = gettime();
  if (bN == 'B') {
    MPI_Barrier(MPI_COMM_WORLD);
  }
  t1 = gettime();
  if (rW == 'R') {
    if (read(fileD, obuffer, iosize) != iosize ) {
      fprintf(stderr, "could not read(), errno = %d\n", errno);
      exit(3);
    }
  } else {
    if (write(fileD, obuffer, iosize) != iosize ) {
      fprintf(stderr, "could not write(), errno = %d\n", errno);
      exit(3);
    }
  }
  tIO = gettime() - t1;
  tBarrier = t1 - tBarrier;
  insertSample(&ioStats, tIO);
  insertSample(&barrierStats, tBarrier);

  if (cS == 'S') {
    rc = lseek(fileD, ((numTasks - 1) * iosize), SEEK_CUR);
  }
  if ( rc == -1 ) {
    fprintf(stderr, "can not lseek(), errno = %d\n", errno);
    exit(3);
  }
}
```

Figure 26. Main Loop of IO Application

The main loop is shown in Figure 26. If barriers are desired, the node will delay until all nodes are synchronized. Next, the I/O operation is performed and if needed, an *lseek()* is done to get the file pointer to the proper place to write the next stripe. The times needed to do the barrier and I/O operations are accumulated. The seek times are not tracked.

5.2.1.5 Read and Write Application Results

Table 8 (Page 1 of 2). Write Performance to /dev/null

Buffer Size (Bytes)	Continuous (no O_SYNC) (MBytes/sec)	Continuous (O_SYNC) (MBytes/sec)	Stripes (lseek, no O_SYNC) (Mbytes/sec)
16	0.42	0.42	0.36
32	0.83	0.85	0.71

Table 8 (Page 2 of 2). Write Performance to /dev/null

Buffer Size (Bytes)	Continuous (no O_SYNC) (MBytes/sec)	Continuous (O_SYNC) (MBytes/sec)	Stripes (lseek, no O_SYNC) (Mbytes/sec)
64	1.70	1.59	1.42
128	3.39	3.40	2.78
256	6.34	6.77	5.76
512	13.53	13.50	11.36
1024	27.12	27.13	23.19
2048	53.07	51.96	45.20
4096	108.90	108.11	87.05
8192	212.95	217.31	184.91
16384	425.27	434.70	370.82
32768	869.15	866.78	694.39
65536	1745.89	1745.64	1455.37
131072	3229.13	3465.16	2953.34
262144	6937.61	6979.96	5926.35
524288	13940.37	13556.64	11144.42

We can see how some of these parameters affect normal AIX by looking at more results for I/O to /dev/null. Table 8 on page 140 provides a long list of buffer sizes using three different types of parameters. As expected, since the marginal cost per byte is zero, performance is determined by the SVC cost in time. This is true over wide range of buffer sizes. The use of O_SYNC does not appear to have a significant effect at the base operating system level. Finally, the need to seek in the striping case causes a rather large decrease in performance.

The /dev/null gives us a baseline from which we can compare other cases; /dev/null is an infinitely fast, zero latency device.

The first GPFS case we consider is the simple, one active client node. Results of writing data to one-disk and two-disk GPFS file systems are shown in Table 9. The disks are 9333 units.

Table 9 (Page 1 of 2). GPFS Performance from Single Active Client

Block Size (Bytes)	O_SYNC	1 Disk FS(MBytes/sec)	2 Disk FS (MBytes/sec)
512	NO	1.12	1.29

Table 9 (Page 2 of 2). GPFS Performance from Single Active Client

Block Size (Bytes)	O_SYNC	1 Disk FS(MBytes/sec)	2 Disk FS (MBytes/sec)
4096	NO	2.26	2.96
16384	NO	2.44	6.19
65536	NO	1.35	1.35
131072	NO	1.35	5.11
262144	NO	3.13	6.27
524288	NO	3.12	6.23

We can see that at small write sizes, the performance is limited by the cost of calling *write()*, while at the large sizes, the performance is limited by the total bandwidth to the disks. With a block size of 262144, the two-disk file system is about twice as fast as the single disk system.

It is not clear what is occurring with the intermediate buffer sizes. The total amount of data that was written was large enough to overflow the local buffers. The 65536 size case is probably near the point where the local buffers cannot hold the data and information must start to be flushed to the physical disks. This will be explored in a later test.

Still, the trend seems clear. To take full advantage of disk performance, I/O must be performed using large buffers.

We now look at the cost of O_SYNC. The O_SYNC flag causes a process to block until the data is safely on the disk. This eliminates any performance benefits that are possible by taking advantage of local buffers and write-behind algorithms. Results for these tests are shown in Table 10.

Table 10. GPFS Performance When Using O_SYNC

Block Size (Bytes)	O_SYNC (MBytes/sec)	no O_SYNC (MBytes/sec)
512	0.0042	1.2229
4096	0.0344	2.2594
16384	0.1344	2.4358
65536	0.4605	1.3499
131072	0.8263	1.3549
262144	2.1242	3.1299
524288	2.5263	3.1209

In this case, we see that the use of O_SYNC causes very serious performance problems when small sizes are written. By the time we get to large block sizes, the performance penalty is not so severe.

In the last simple I/O write test, shown in Table 16, we look at the cost of updating all the metadata structure when a file is created. To observe this cost, we write a file twice: the first time it is created and the second time it is overwritten. Table 11 shows that it is considerably less expensive, in terms of time, to overwrite a file.

Table 11. GPFS Performance of Create Vs. Overwrite

Block Size (Bytes)	Create (MBytes/sec)	Overwrite (MBytes/sec)	1st Time Penalty
4096	0.03	0.03	0.0%
65536	0.50	0.54	7.5%
131072	0.93	1.09	14.7%
262144	1.55	2.13	27.2%
524288	1.87	2.47	24.3%

These tests were run with the use of O_SYNC, so the cost of the actual I/O would be the same in the first and second run. The cost of creating a new file can be relatively large, so if your application can re-use many files rather than deleting and creating them, that possibility should be explored.

Now we consider I/O read performance on a single client node. Here we consider a number of buffer sizes and measure the performance of a one- and two-disk file system. Table 12 shows how well the cold cache read times scale when the second disk is added.

Table 12. GPFS Read Performance from Single Active Client

Block Size (Bytes)	1 Disk FS (MBytes/sec)	2 Disk FS (MBytes/sec)
128	0.26	0.30
4096	3.40	5.48
65536	3.25	3.14
131072	3.18	6.03
262144	3.17	6.01
524288	3.18	6.05

Again, in this test we see an odd dip in performance at an I/O size of 65536, although the dip is not as pronounced as in the write case. As expected, with very small sizes such as 128 bytes, the performance is rather low due to the system call overhead. GPFS is quickly able to reach the I/O limitation of the physical device and can sustain that level of performance. This can be a very large benefit for existing AIX utilities that use moderate-sized I/O buffers. This good read performance should allow most applications to exploit the parallel disk capabilities of GPFS with no alteration.

Now we move on to multi-node tests. In these tests, file systems constructed with two SCSI 2 drives are used.

This time we will look at the read cases first. Table 13 shows what happens when we try to stress the GPFS file system with parallel reads and a cold cache on the nodes.

Table 13. GPFS Read Performance from 4 Active Clients (Cold Cache)

Block Size (Bytes)	2 Disk FS (MBytes/sec)
65536	7.02
262144	6.91

In these cases, GPFS can easily keep up with the physical disks. Few different cases were measured as it is difficult to force the cache to be empty. The file systems must all be unmounted and then remounted between each case.

Table 14. GPFS Read Performance from Active Clients (Warm Cache)

Nodes	Block Size (Bytes)	MBytes/sec
1	65536	37.59
2	65536	80.34
4	65536	163.14
8	65536	346.29

When we look at the warm cache cases in Table 14, we see that warm cache reads require little or no communication. Once the data is in the nodes′ caches, it can very quickly be transferred to the applications. This should allow GPFS to perform with very low latency if it is used to distribute system files and project files to a number of nodes in an SP.

Both of the previous tests read the file in a continuous order. This allows the read-ahead algorithms to determine that it would help performance to transfer extra information to the node as there is a good chance it will soon be used. If the read pattern is not sequential, as would happen if a multi-node parallel process decided to have each node read a strip of the file, performance drops because the read-ahead facilities do not figure out what is happening. This is shown in Table 15.

Table 15. GPFS Read Performance with 4 Active Clients (Cold Cache)

Block Size (Bytes)	Consecutive (MBytes/sec)	Stripes (MBytes/sec)
4096	1.34	0.88
131072	3.60	2.64
262144	3.28	3.29
524288	3.28	2.68

While the use of non-sequential access patterns can cause extra data to be transferred to a node or can disguise sequential reads and reduce the effectiveness of read-ahead, even worse performance problems can occur if writes are done in striped rather than sequential access patterns. Table 16 shows an example of the difference between consecutive and striped access patterns.

Table 16. GPFS Update Performance from 4 Clients Sharing a File Block

Access Pattern	MBytes/sec
Continuous	3.01
Striped	0.89

The problem with writes is worse than for reads because of the way GPFS enforces strict semantics. When two nodes try to modify the same block, one of the nodes must wait until the other node releases the block. For small block sizes, this can cause performance-robbing communication between nodes as they negotiate an order for updates. In this case 4 nodes wrote 65535 bytes pieces of a file that was stored in a GPFS file system with a 262144 byte block size. This caused the 4 nodes to perform up to 4 negotiations for each file block, reduced GPFS caching benfit, and increased the number of disk accesses.

Fortunately, for most applications the more general write operation can take advantage of write-behind capabilities. This works a lot like read-ahead but in reverse. The extra performance that write-behind provides is shown in Table 17 on page 146.

Table 17. GPFS Write Performance from 4 Active Clients

Nodes	Block Size (Bytes)	MBytes/sec
1	65536	12.25
2	65536	9.18
4	65536	11.75
8	65536	28.47

Though the peak performance does not seem very smooth with an increase in nodes, it can be seen that the instantaneous write performance can be much higher than the physical transfer capacity of the disk devices which in this case is about 6MBytes/sec. The information is stored in buffers on the nodes and transferred to the disks at a later time. Of course there are limits to the amount of data that can stored locally, but for many applications, this local buffer can greatly increase the I/O performance of applications. As with the read-ahead case, these buffers allow unaltered applications such as standard AIX utilities to use most of the performance of GPFS.

Table 18 shows the cost of using the O_SYNC flag on multiple nodes. As expected, by forcing each write to bypass the local buffers and store the data on the disks, performance is lost. Fortunately, as nodes are added, total I/O performance rises. Still, you will only want to use this flag if it is needed because the performance falls by more than a factor of 10.

Table 18. GPFS Write Performance with O_SYNC from 4 Active Clients

Nodes	Block Size (Bytes)	MBytes/sec
1	65536	0.47
2	65536	0.76
4	65536	1.01
8	65536	1.14

All of these examples have used obvious buffer sizes that are powers of 2. When I/O is done with strange buffer sizes, performance can suffer in odd ways. As an example, consider what happens when multiple nodes write to a file using buffers of size 36703 bytes. This is shown in Table 19 on page 147. As with the other write examples, there is a good deal of negotiation that must be done to guarantee the semantics

of the write. With odd buffer sizes, it is hard to predict how the write tokens will flow.

Table 19. GPFS Write Performance with 4 Clients and 36703 Byte Buffers

Nodes	MBytes/sec
1	23.70
2	7.95
4	36.75

Last, we look at the effect of using MPI barriers to synchronize I/O operations across many nodes. In Figure 27, we see an example report from sample test runs with and without MPI barriers. At first glance, it appears that synchronizing the I/O operations hurts I/O performance because the performance drops from 3.01MBytes/sec to 1.43MBytes/sec. In the barrier test, the MPI_Barrier function takes more wall time than the write operation (10.25e-2 vs. 8.03e-2):

```
**** Ran from 1997/10/02 18:23:35 to 1997/10/02 18:24:19 on 4 nodes. ****
./pio 500 65516 0 0 W N C N N /gpfs53/junk
 Calibration Overhead (removed from stats)
 AVE=5.630297e-06, STDEV=2.625584e-05, MIN=5.224982e-06, MAX=2.532625e-03

 Barrier
 AVE=6.181100e-06, STDEV=9.799038e-07, MIN=5.674985e-06, MAX=1.057499e-05
 IO
 AVE=8.709669e-02, STDEV=2.239099e-01, MIN=1.097025e-03, MAX=9.233720e-01

Wall Time = 043.5591  User Time = 000.0200  System Time = 000.6400
wall time - (user + system time) =  42.8991 (other system time)
This process got  1.515% of the available CPU time
Total data rate was   3.0081Mbytes / second

**** Ran from 1997/10/02 18:24:35 to 1997/10/02 18:26:07 on 4 nodes. ****
./pio 500 65516 0 0 W B C N N /gpfs53/junk
 Calibration Overhead (removed from stats)
 AVE=5.305732e-06, STDEV=7.892098e-07, MIN=5.224982e-06, MAX=4.487499e-05

 Barrier
 AVE=1.025274e-01, STDEV=2.121575e-01, MIN=1.302000e-04, MAX=1.969075e+00
 IO
 AVE=8.034483e-02, STDEV=1.839963e-01, MIN=1.118950e-03, MAX=7.587111e-01

Wall Time = 091.4426  User Time = 049.9000  System Time = 000.6600
wall time - (user + system time) =  40.8826 (other system time)
This process got 55.292% of the available CPU time
Total data rate was   1.4329Mbytes / second
```

Figure 27. Use of MPI_Barrier to Synchronize Writes

Since the total data rate is calculated using wall time it is easy to see why the data rate falls by more than 50%. On the other hand, 0.1 sec for an *MPI_Barrier* operation for four nodes is quite large. As an example, Figure 28 on page 148 shows that when the I/O is done to /dev/null, a device with very predictable and high-performance characteristics, the barriers only take 1.48e-4 seconds.

```
**** Ran from 1997/10/02 18:24:35 to 1997/10/02 18:26:07 on 4 nodes. ****
./pio 500 65516 0 0 W B C N N /gpfs53/junk
 Calibration Overhead (removed from stats)
 AVE=5.305732e-06, STDEV=7.892098e-07, MIN=5.224982e-06, MAX=4.487499e-05

 Barrier
 AVE=1.025274e-01, STDEV=2.121575e-01, MIN=1.302000e-04, MAX=1.969075e+00
 IO
 AVE=8.034483e-02, STDEV=1.839963e-01, MIN=1.118950e-03, MAX=7.587111e-01

Wall Time = 091.4426  User Time = 049.9000  System Time = 000.6600
wall time - (user + system time) =  40.8826 (other system time)
This process got 55.292% of the available CPU time
Total data rate was   1.4329Mbytes / second
```

Figure 28. Example Using Barriers When Writing /dev/null

The extra time needed to complete the MPI_Barrier is consumed someplace on the nodes. Given the /dev/null results, we can assume the extra time occurs due to GPFS daemons which cause some nodes to be delayed. If your application uses threads and allows the I/O operations to complete on an independent thread, you might be able to reduce the effects of highly synchronized I/O operations. Though some time and performance was consumed by the barrier function, most of the performance went to GPFS-related daemons.

5.2.1.6 Read and Write Application Summary

This section touches on a wide variety of access patterns, block sizes and other topics such as the use of synchronized access. Hopefully these examples give you ideas on ways to get the best performance possible from your system. In most cases, you will probably have to experiment to determine the optimal methods for your application′s access patterns.

When coding for I/O intensive applications, the issues fall in three main areas:

1. Issues that are common to many file systems and applications
2. Issues that are unique to parallel applications

3. Issues that are unique to parallel applications that use GPFS

Keep the following suggestions in mind when dealing with these areas.

In the first category, we consider the normal ideas such as using *read()* and *write()* over their stream-based counterparts *fread()* and *fwrite()* in most cases. Use large block sizes when possible. Keep the buffers aligned. Do not use *O_SYNC* or similar features that reduce the possibility of low-level parallelism between the I/O and other, CPU-intensive work. Overwrite existing files when possible to reduce the manipulation of metadata, and so on.

In the parallel arena, with most parallel file systems it has been better to have one node do the I/O and redistribute the information across nodes using MPI or some similar function. Most distributed file systems do not handle cases of multiple, synchronized client requests very well. Also, it is often the case that the distributed file systems either cannot use the SP switch for fast transfers, or even if the switch is available, it is not used in an efficient manner. As an example, older NFS implementations were limited to 8K transfers.

The combination of parallel jobs on an SP with GPFS opens new opportunities as well as hazards. In a high I/O volume application that uses GPFS, it is important to use large, well-aligned buffers when possible. If small block records must be used, try to perform all I/O from one node or a small set of nodes with full block access. This will reduce the time needed by GPFS to pass the write tokens from node to node.

When reading data, try to have as many clients active as possible. Also try to read the data in continuous chunks rather than stripes. As an example, it is better to have 4 nodes read data in the pattern *1111222233334444*, rather than have each node skip around in a pattern like *1234123412341234*. The first pattern is more likely to trigger the read-ahead algorithms and less likely to transfer unused information to the nodes in the form of read-ahead blocks or partial blocks. Also, the first approach saves *SVC* calls because the *lseek()* function is not needed between reads.

Exercise care when partitioning a task across many processors. For I/O-intensive tasks, it makes sense to take extra time to make sure the boundaries used to partition a data set across nodes are located on block sized offsets in the file. The *stat()* function call on a file will indicate the block-size to use. This can save much time, especially if many of the I/O operations will be writes.

Finally, remember that these measurements were made on relatively slow SP components. When faster processors are used, the relative percentage of resources that go to GPFS decreases and the trend favors the applications.

5.2.1.7 Additional Suggestions from Discussions with the Developers

Here is some additional information from the GPFS developers which may help you get the best possible performance from GPFS. These situations are difficult to show in small, simulated applications.

When different parts of a data block are written by different nodes, the whole block of data must be written from the first node to disk. Then the modified block is read to the second node so the modifications can be done. If you know that some other node will soon try to modify a block, it might make sense to force that block to the VSD server with a *fsync()* or through use of the *O_SYNC* flag. This should only be done if you are very certain about the access patterns, as these are expensive operations.

The GPFS file system is actually implemented as daemons that operate on the same node as the applications that are using the file systems. The GPFS processes generally do not need too much CPU resource, but they can consume other resources such as memory, network bandwidth (important to communication-intensive MPI applications), and cache memory slots. If your applications are not floating point math-intensive, you might getter better I/O performance with SMP nodes.

The current GPFS implementation is not able to detect applications that are reading at constant strides in a file. Future versions should be better able to detect this condition and use a form of read-ahead to increase performance.

5.2.2 Summary and Suggestions

In many cases, GPFS will provide very good performance with no special effort from the programmer or system administrator. These cases occur with data is read in large chunks that correspond with file system block sizes, and few metadata changes are required.

Some traditional lore about the best way to achieve high I/O performance in an MPI environment (which read the data in one node and use MPI to push the data to other nodes) does not hold true with GPFS. In most cases GPFS transfer rates increase with the number of active client nodes. If data can be read or written in large chunks that do not overlap, it is often better to let each task do its own I/O. This can be the case even if moderate amounts of data must be read by all the

tasks before processing can start. Bottlenecks associated with file systems such as NFS are not a large concern in many SP environments which use GPFS.

The places where special attention is needed include applications that perform many small writes in areas that might overlap, or when metadata is changed frequently. These operations force file systems that maintain robust semantics such as GPFS to do extra work. On a totally local file system such as JFS, the communication associated with that work is small, but in a distributed file system such as GPFS, that cost can be significant if care is not used.

In designing new applications that will use GPFS, try to perform all metadata operations in one node.

5.3 Parallel Sorting of Large GPFS Files

I/O is an abbreviation for input and output of data. In a general context, I/O refers to the movement of data between different levels of a memory hierarchy such as hardware registers, caches, main memory, disk, tape, remote network storage devices and so on. In this section, we restrict our discussion of I/O to the movement of data between memory and disk (either local or remote).

5.3.1 I/O Requirements of a Parallel Application

The need for I/O arises at different stages during the execution of a program as illustrated in Figure 29 on page 152. These phases are:

- Initialization
- Computation
- Termination

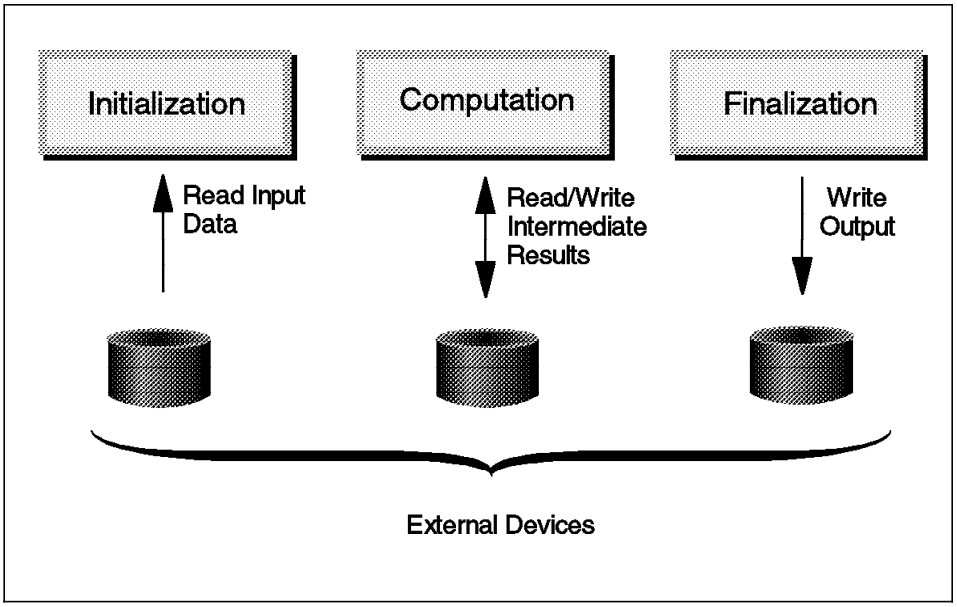


Figure 29. I/O Requirements in an Application

During the *initialization* phase, the application needs very high bandwidth transfer of data from external devices to build application-specific data structures which are used later during the computation phase. During the *computation* phase, the need for I/O arises due to the following requirements:

- Checkpointing
- Store and fetch intermediate results

Checkpointing involves saving the state of the application at a particular instance in time so that it can be restarted from that state in the case of catastrophic failures. For large applications, checkpointing is an I/O-intensive operation.

The need to *store* and *fetch* intermediate results arises when the application is too large to fit into the main memory. In this case, it is necessary to move the partial state of the application onto external devices so that the processor can work on other parts of the program which need to be completed. The size and frequency of storing and fetching results can vary from small amounts at high frequency to large amounts written sporadically.

During the termination of the application, the requirements for storing the final output for later use can be as massive as those of the initialization phase.

Now that we have established that some programs need large amounts of data residing on external I/O devices, it may be useful to know why we need to transfer the data at very high speeds between the main memory and the external I/O devices. This is due to the increase in the power of individual processors by several orders of magnitude as compared to the improvement in external I/O devices. The ability to combine several uniprocessors has widened the gap even further. The combined I/O bandwidth on some of these supercomputers is several orders of magnitude less than that of their combined processing capacity. It is precisely this type of disparity which has led some people to declare that we are in the age of “I/O crisis.”

Some of the approaches which are used to close the gap between processing speeds and I/O speeds are based upon two popular models:

- The parallel disk I/O model
- The parallel network I/O model

5.3.1.1 Parallel Disk I/O Model

Both the parallel disk I/O model model as well as the parallel network I/O model are based on the theory that transfer capabilities of slow I/O devices can be leveraged to achieve a higher sustainable transfer rate locally at a processor than would otherwise be possible with a single disk alone.

For example, as shown in Figure 30 on page 154, eight disks with 10MBytes/sec transfer capability may be used to attain 80MBytes/sec transfer rate from disk to the memory.

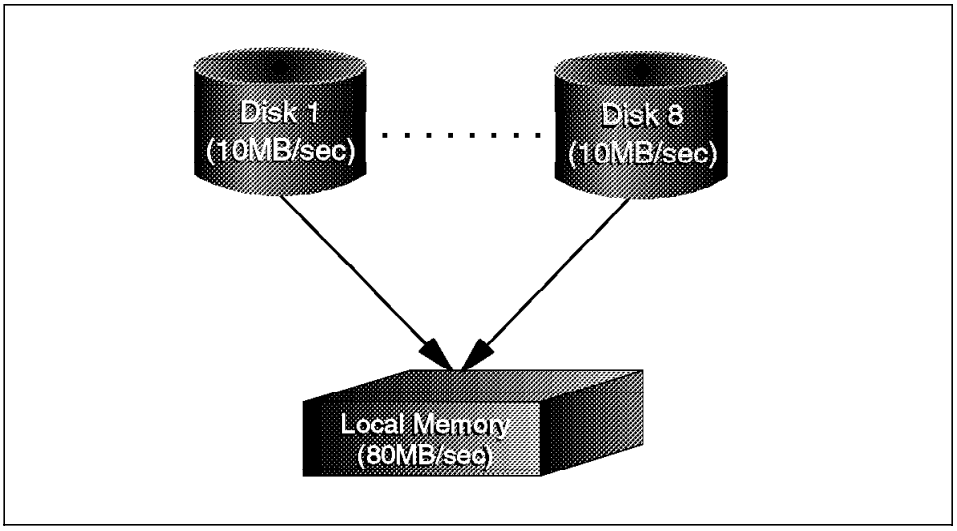


Figure 30. Parallel Disk I/O Model

5.3.1.2 Parallel Network I/O Model

In the parallel network I/O model, a file is striped across multiple processor nodes as well as the disks within a node.

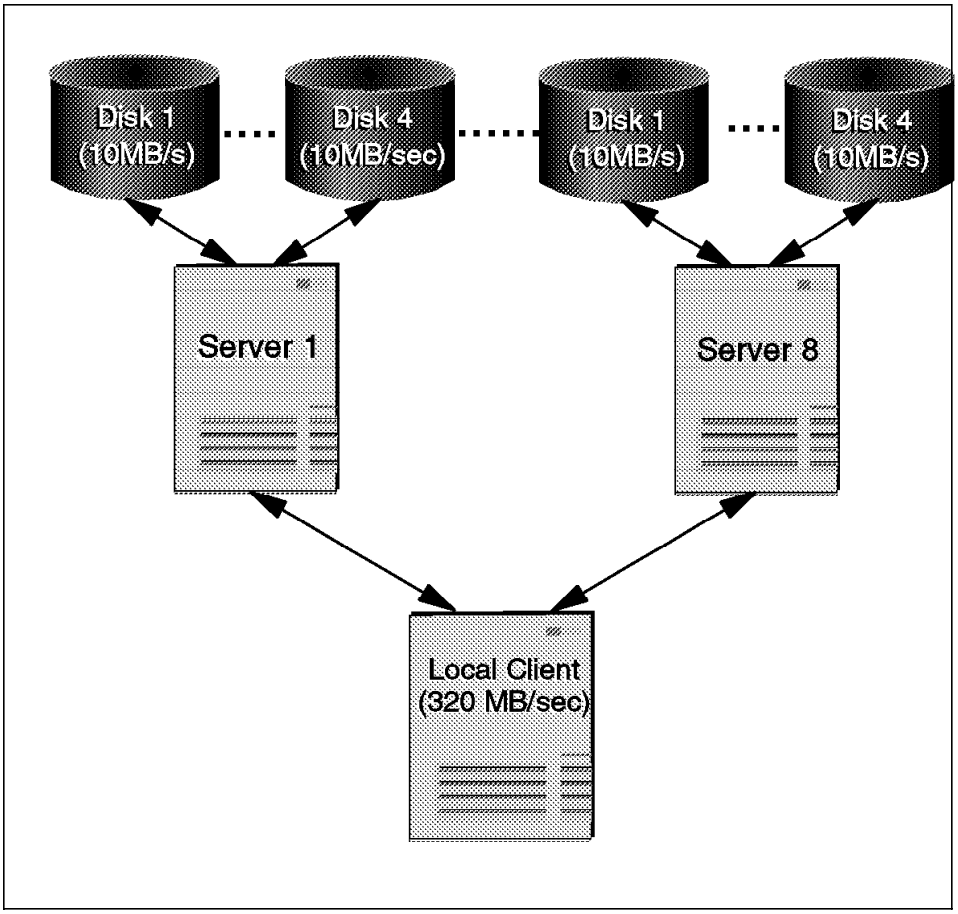


Figure 31. Parallel Network I/O Model - 1 Client and 8 Servers

For example, as shown in Figure 31, it may be possible to achieve a 320MBytes/sec transfer rate at a local processor when the file system is striped across eight nodes with four disks on each processor.

Additionally, each of the four different clients could be reading from a separate processor as shown in Figure 32 on page 156. In other words, four simultaneous accesses to the file system can be handled.

General Parallel File System (GPFS), as shown in Figure 53, presents the parallel network I/O model to the user.

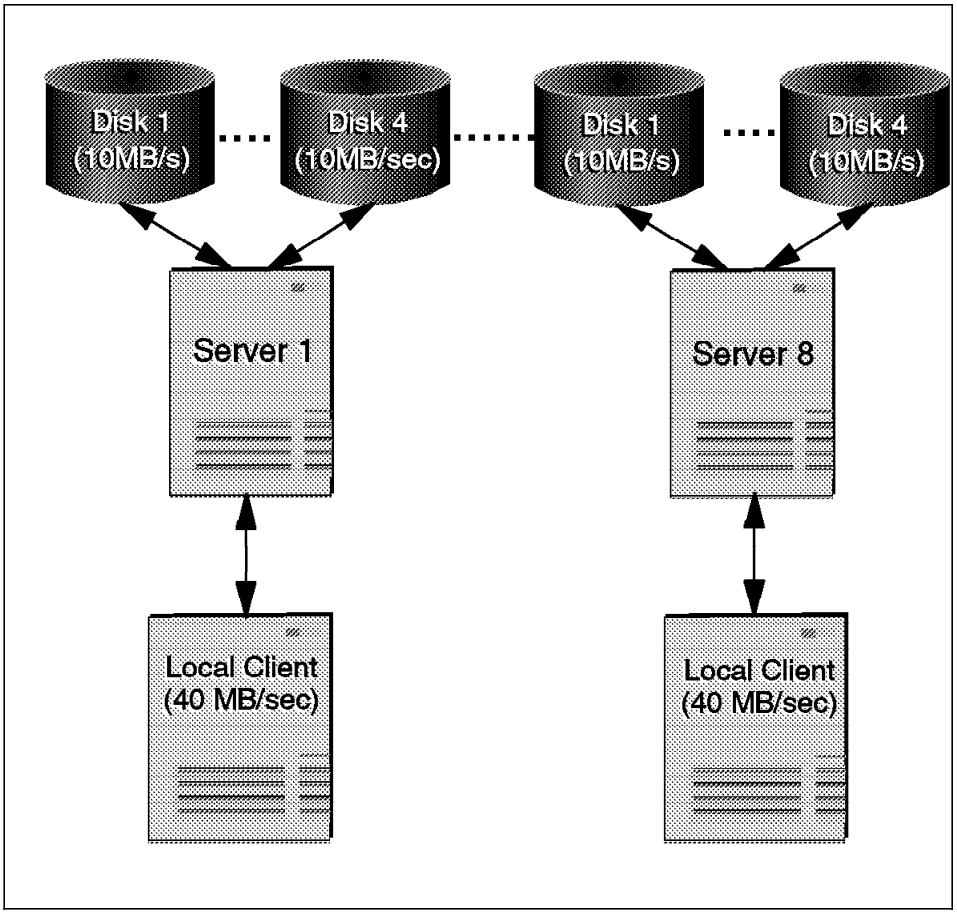


Figure 32. Parallel Network I/O Model - 8 Clients and 8 Servers

This model stripes data transparently across multiple disks and processors and presents a UNIX file interface to application developers. This model enables the development of high performance applications with inherent I/O parallelism by making use of GPFS.

5.3.2 Goals of the Current Study

This section describes an experiment we carried out to explore the impact of JFS, NFS, and GPFS on the performance of an I/O-intensive application such parallel file sorting. Two main parameters that are of interest in this study are the number of application tasks (also called clients) and the number of I/O servers which cooperatively manage the I/O system. For example, in JFS there is only one processor which performs all the I/O tasks. In GPFS, multiple processors can share the responsibility of managing the I/O subsystem.

The rest of the this chapter is organized as follows. In 5.3.3, “Parallel Sorting Application” on page 157 a parallel sorting application is introduced. Some details of the parallel sorting algorithm are discussed in 5.3.4, “Parallel Sorting Algorithm” on page 158. A summary of its I/O requirements are outlined in 5.3.5, “I/O Requirements of a Parallel Sort Algorithm” on page 162. The measures of performance and the controllable parameters which were used this experiment are discussed in 5.3.8.1, “Measures of Performance” on page 165. Final results are presented in 5.3.10, “Results of Experiments” on page 169, along with some conclusions.

5.3.3 Parallel Sorting Application

Sorting of large data sets is done frequently in database management and results in a variety of report generation tasks. In large mass-storage devices, large data files are kept sorted with respect to certain fields so that on-line queries for data can be processed very quickly. Sorting large data sets is very I/O- and data-intensive. Performance of the sort utility provided by a transaction management system forms one of the fundamental measures of processing power for the sorting mechanism. Some of the factors which affect the performance of a sort utility are:

- Sorting algorithm
- Number of disks and processors
- Number of application tasks/processors
- System parameters

Several sorting algorithms with varying time complexity and varying I/O needs have been proposed in the past. In this study, a variation of the well-known quicksort algorithm known as *sample sort* is used to measure the I/O performance levels which can be achieved when GPFS is used. Varying the number of processors used to manage the I/O subsystem and the disks attached to each of these processors located at each of these processors can result in different levels of I/O performance. As more processors are put to work on the sorting problem, it is expected to take less time to complete the job. Finally, certain system parameters also impact the performance of the sorting algorithm. For example, a mismatch between the average block size of the file system and the application may have a drastic impact on the performance of an application.

5.3.4 Parallel Sorting Algorithm

Parallel sorting application consists of several tasks operating concurrently and cooperatively to sort a large dataset. Each of these tasks executes the steps shown in Figure 33. During the execution, each task generates results which are shared by other tasks. The file system is used as a medium to share some of the partial results among the tasks. Initially, all the data which needs to be sorted is assumed to be located in a large sequential file accessible to all the tasks. The file system in which the file resides can be one of the following:

- Journal File System (JFS)
- Network File System (NFS)
- General Parallel File System (GPFS)

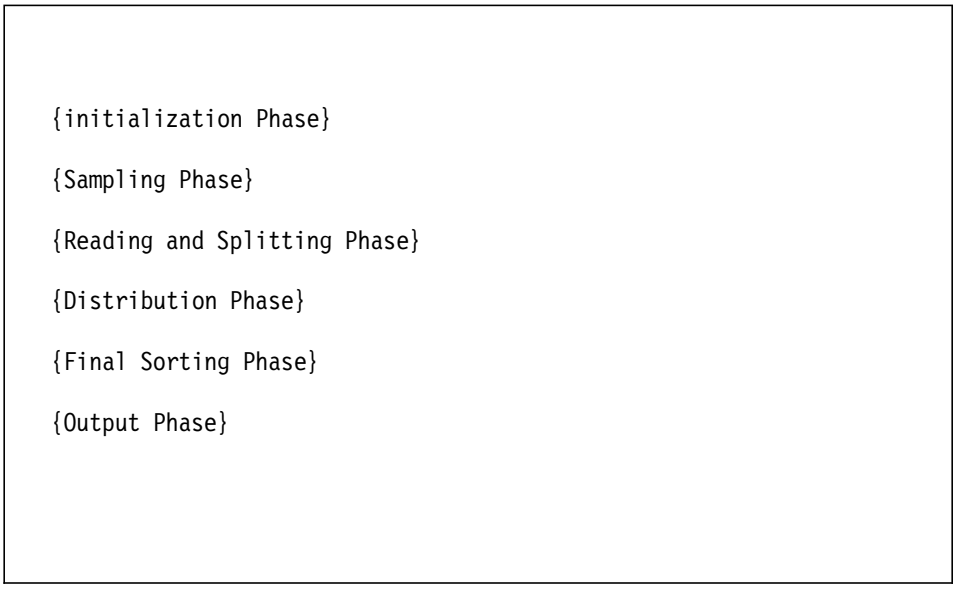


Figure 33. Parallel Sorting Algorithm

During the *initialization* phase, each of the tasks queries the system to obtain the operating parameters (shown in Figure 34 on page 159). Some of the important parameters are the number of I/O servers and the number of tasks participating in the application. The operational details of the parallel sort algorithm are explained in Figure 35 on page 161 and Figure 36 on page 162, using a simple illustration.

```
{Initialization Phase}


(1)  Query environment

       N = Number of tasks            = 2
       S = Size of file               = 20
       M = Available memory size      = 5

(2)    Compute

       FT = Total number of fragments          = S/M = 20/5 = 4
       T = Number of splitting keys            = FT -1
       FA = Average number of partitions/task  = FT/N
       K  = Sampling keys                      = 2

(3)    Task identification                     = 1,2


(3)    For task 1:

       Allocation Range                = 1 through 10
       Allocated fragments             = 1 and 2
       Range of fragment 1             = 1 through 5
       Range of fragment 2             = 6 through 10

(3)    For task 2:

       Allocation Range                = 11 through 20
       Allocated fragments             = 3 and 4
       Range of fragment 3             = 11 through 15
       Range of fragment 4             = 16 through 20
```

Figure 34. Initialization Phase of Parallel Sorting Algorithm

Each record in the file to be sorted consists of a key on which the file needs to be sorted and some data. It is assumed that the file is too large to fit into the combined memory of all the processors. This forces the program to do some I/O during the course of its execution to store and fetch intermediate results. The parameters file size and memory size available to the application are used to divide the file into a number of fragments where each fragment is just big enough to fit into the memory without causing any paging activity. During *reading* and *splitting* phases, each processor brings one fragment at a time into the

memory from the file system to perform the splitting as illustrated in Figure 36 on page 162. The allocation of fragments to a processor is arbitrary. For the sake of simplicity it is assumed that the first *FT/N* fragments, where *FT* is the total number of fragments and *N* is the number of tasks, are allocated to the first processor and the next *FT/N* fragments to the next processor and so on.

Once the respective allocation range is determined, it is time for each of the processors to sample its data for potential candidates as the *splitting* keys. Initially, all the processors sample their allocation of the file at random locations for the key values. The sample size is relatively small and in the range of 64-1024 depending on the size of the file. The sampling results are mutually exchanged among the participating tasks and sorted as shown in Figure 36 on page 162. From the sorted collection, each task selects the first ***T*** *splitting keys* with a spacing of sample size between them. These splitting keys are used to split each fragment into *FT* partitions.

During the *distribution* phase, each task stores the partitioned fragment in a temporary file and saves the pointers to the partitions within a fragment and exchanges this information with other tasks. After processing all the fragments assigned to itself, the next step for each of the application tasks is to process the partitions. Again, the allocation of partitions to the application is arbitrary. Unlike the fragments each partition is scattered across the entire file space. The task assigned to a partition gathers all the pieces by following the pointer information exchanged during the splitting phase. Once all the pieces of a partition are collected together, the partition is sorted in memory and output to a permanent dataset. These steps are illustrated using a simple dataset in Figure 36 on page 162.

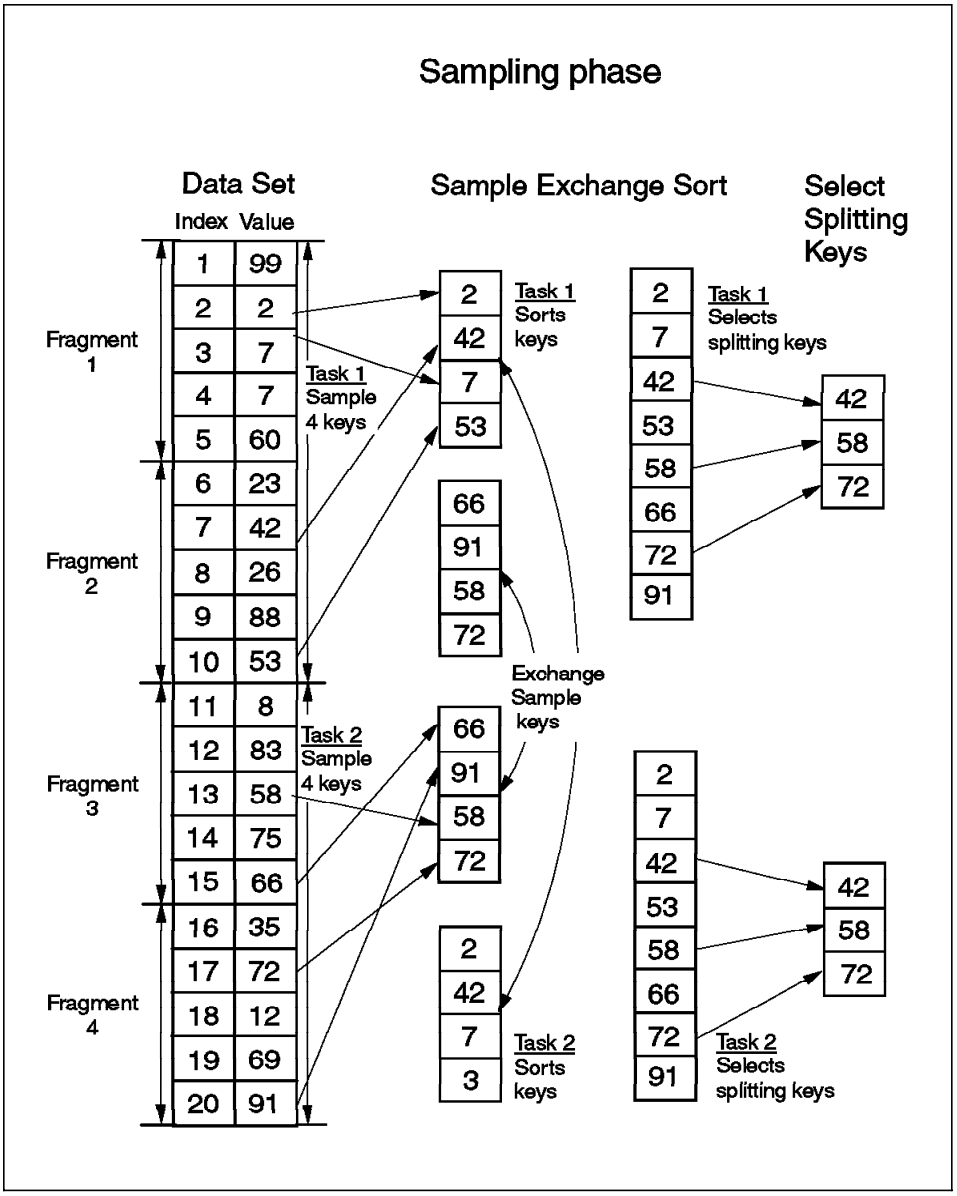


Figure 35. Sampling Phase of Parallel Sorting Application

For example, as shown in Figure 36 on page 162, a fragment is split into FT partition or buckets (4 in this case). The first FA partitions (2 in this case), where FA is the average number of partitions, are intended for processor 1. The next FA partitions (in this case) are intended for processor 2, and so on. Since only one fragment can be stored in the memory of a given processor at a time, the partitions obtained from its

respective fragments are stored in a large file. Clearly, the I/O requirements are intense during this phase.

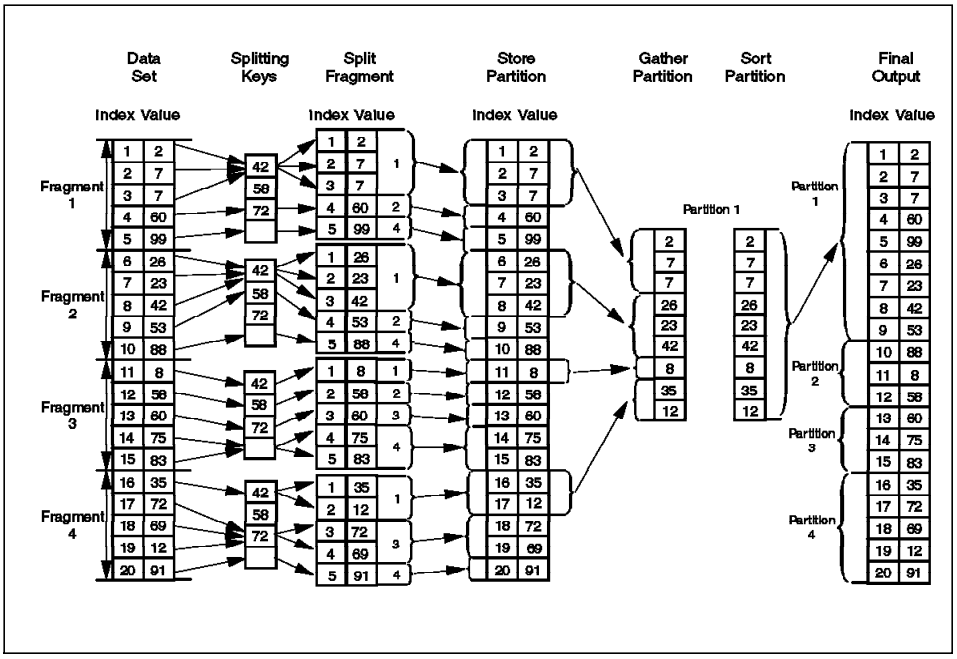


Figure 36. Reading, Partitioning, Gathering, Sorting and Output Stages

5.3.5 I/O Requirements of a Parallel Sort Algorithm

The I/O requirements in the parallel sort algorithm are summarized in Figure 37 on page 163. During the initialization and sampling stages, the I/O requirements are small and the access pattern is random.

During reading and splitting phases, fragments are read and partitions are stored. I/O requirements during these phases are very large and sequential. During gathering stage, each partition is collected from different parts of the temporary file, the I/O requirements are medium, and the access in mostly sequential. Finally, during the final output phase the I/O requirements are large and sequential in nature. For example, in order to sort a 250MB dataset, 250MB of data is read from the input dataset and 250MB of data is written to an intermediate file. During the gathering phase, 250MB of data is read from the intermediate file and during the final output phase 250MB of data is written to the permanent dataset. Altogether, 500MB of data is read and 500MB data is written during the execution of the parallel sort application. This

measure is used to compute the aggregate bandwidth achieved by the application.

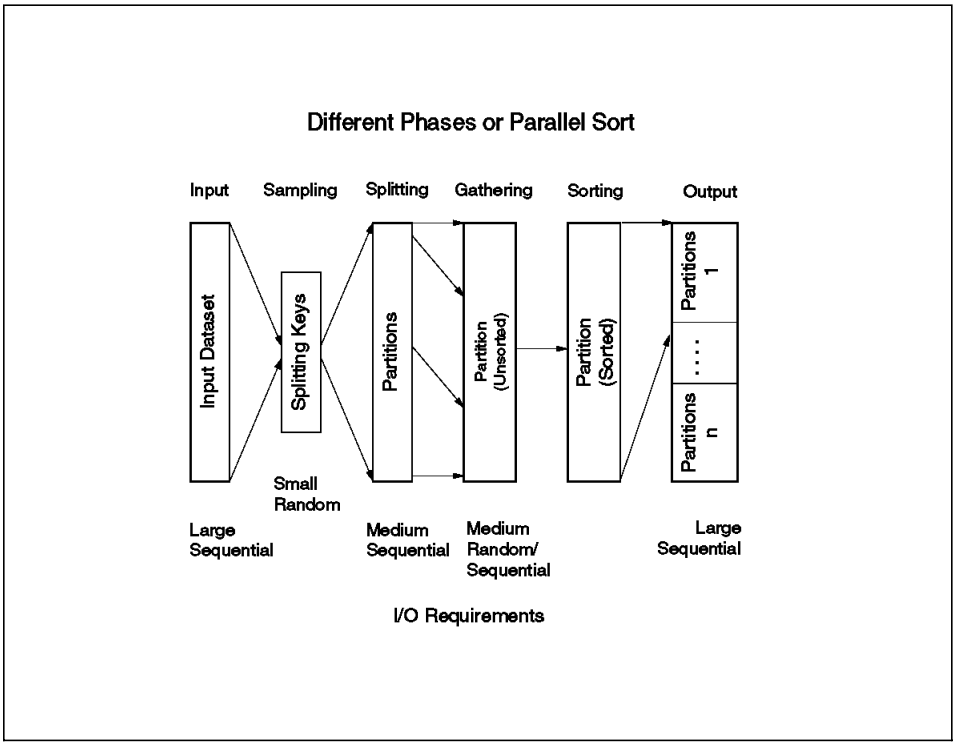


Figure 37. Summary of I/O Requirements for Parallel Sorting

5.3.6 GPFS Programming Interfaces

Applications access GPFS data using standard AIX system calls and libraries. Support for large libraries is provided through the use of AIX 64-bit forms of these libraries. See the AIX documentation for details.

GPFS is designed to be compliant with the file system calls specified in the X/Open standard with the following exceptions:

1. The following memory mapped calles are not supported in this release
   - mmap
   - munmap
   - msync
   - shmat

The following calls are used to perform I/O in the parallel sorting application:

- fopen
- fseek
- fread
- fwrite
- fclose

5.3.7 System Configuration

The configuration of the RS/6000 Scalable PowerParallel (RS/6000 SP) system used to conduct the parallel sorting experiment, as shown in Figure 38 on page 165, consists of the following hardware components:

- RS/6000 SP with 16 thin nodes (66 MHz model 390s)
- High Performance Switch
- 7 SSA drives attached to nodes 5 and 6
- 3 9333 drives attached to 3 and 4

The software components of the configuration are:

- AIX 4.2.1
- Parallel Systems Support Programs 2.4
- IBM Parallel Environment for AIX 2.3
- General Parallel File System 1.1
- RVSD 2.1.1

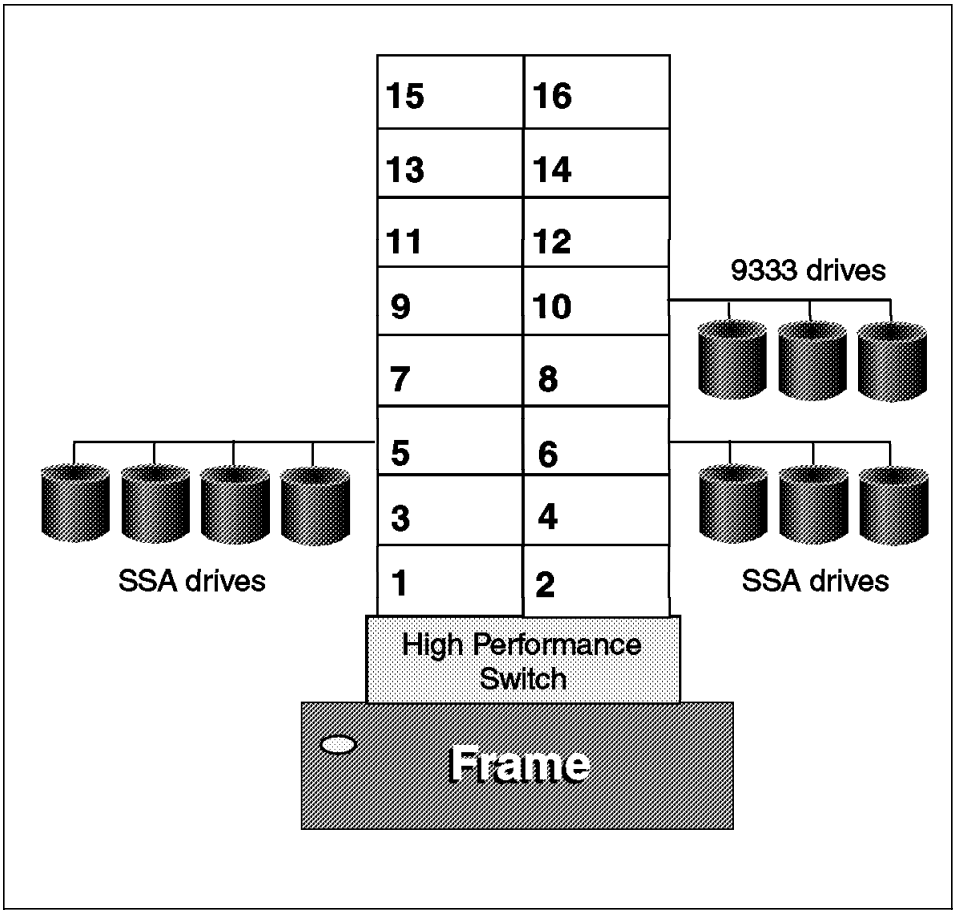


Figure 38. The RS/6000 SP Configuration Used to Run Parallel Sort

5.3.8 Description of Experiments

In this section the measures which are of some interest in the parallel sorting application and the parameters which may affect these performance measures are discussed.

5.3.8.1 Measures of Performance

The following measures of performance, as defined, are used:

Write Time The application blocks when it issues a file system call such as *fwrite* until the data is copied out of application buffer. This is the total time spent by the application waiting for the I/O subsystem to take control of the data when a call such as the *fwrite* is issued. The application need not wait until all the data is written to the disk and is free to proceed as soon

as the system takes control of the data. This mode of operation is called *asynchronous I/O* and is expected to improve the performance of the application by overlapping the I/O and computation.

Total Write Bandwidth This measure represents aggregate write bandwidth achieved by the system and is computed by dividing the total amount of data written by average total write time per task.

Read Time The application blocks when it issues a file system call such as *fread* until the data is available in the application buffer. This is the total time spent by the application waiting for the I/O subsystem to complete the *fread* operation.

Total Read Bandwidth This measure represents aggregate read bandwidth achieved by the system and is computed by dividing the total amount of data read by the average total read time per task.

Computation Time This measure represents the time spent by the application in non-I/O activity, which includes compute time, and is computed as the average of the non-I/O times spent by each task.

Total Completion Time This is the time to complete application and is equal to the summation of read time, write time, and non-I/O time.

5.3.8.2 Parameters

Several factors may influence the performance of the Parallel Sorting program. These factors are:

Type of File System Used The following file systems are used in this study:

- Journal File System (JFS)
- Network File System (NFS)
- General Parallel File System (GPFS)

Number of Server Nodes Used in the File System This parameter represents the number of processors used to serve the file system. For example, for the configuration with one node and with seven SSA drives, there are seven disks and only one server node.

Number of Disks Per Node This parameter determines how many disks are used on a node. In the configuration used for this study, six SSA drives are on node 5. When the I/O subsystem is configured to use only one node and 6 SSA drives, then the I/O server is configured to run on node five and is allocated all six drives.

GPFS uses the IBM Recoverable Virtual Shared Disk technology for device management. IBM Recoverable Virtual Shared Disk provides transparent access to remote storage devices. In this study it is assumed that only one disk is used per Virtual Shared Disk. This ensures that all the striping on the Virtual Shared Disks is done by GPFS. For the nodes without any SSA drives, internal drives are used.

Tasks/Processors in the Parallel Sorting Application Number of processors or processes working in the parallel sorting application.

Size of the Sort Data Set This is the size of the data set which is sorted by the parallel sort program. As discussed in 5.3.5, "I/O Requirements of a Parallel Sort Algorithm" on page 162 for a dataset of size M bytes, total I/O requirements in this application are 4M bytes (2M bytes are read and 2M bytes are written).

5.3.8.3 Summary of the Parameters

The parameters and their values used in this experiment are summarized in Table 20.

Table 20. Summary of Parameters and Their Values Used in the Sort Experiment

Parameter Name	**Values used**
File system type	JFS, NFS, GPFS
Processors in File system	1,4,8,12
Tasks/Processors in the parallel sort program	1,4,8,12
Number of SSA disks per node	1, maximum allowable
Size of the input data set	32MB, 128MB, 256MB

Some of the combinations of these factors are not applicable. For example, multiple servers are not used to serve the JFS file system.

5.3.9 Running Parallel Programs Using IBM Parallel Environment

IBM′s Parallel Environment provides facilities to write, debug, start, and monitor parallel applications which use message passing as the main vehicle for communication and synchronization. A summary of some of the facilities used in the parallel sort application is given here. For more details of IBM′s Parallel Environment refer to the relevant documentation listed in Appendix H, "Related Publications" on page 223.

5.3.9.1 Starting Parallel Tasks

In order to start a parallel application, the information listed in Table 21 is provided to the program poe supplied by IBM Parallel Environment for AIX.

Table 21. Command Line Parameters to Start Parallel Jobs Using POE

Description	Command line parameter	Allowed values
Number of tasks/clients	-procs	1 to 12
Communication device	-euidevice	en0, css0
Communication library	-euidevice	us for css0 ip for en0 and css0
List of nodes used	-hostfile	Path of any file containing a list of nodes.
Parallel program		gpfs_sort

A sample run made in this experiment using 12 tasks to sort a file of 256MB residing in a GPFS file system can be made by typing the following command on the screen shown in Figure 39.

```
poe gpfs_sort       gpfs_sort      256000000       \
                    /gpfs/file.input               \
                    /gpfs/file.output              \
                    -procs         12              \
                    -euidevice     en0             \
                    -euilib        ip              \
                    -hostfile      hostlist
```

Figure 39. Sample Command to Start a Parallel Application Using poe

Once all the tasks are started using the IBM Parallel Environment, they can use the message passing libraries provided by the IBM Parallel Environment to exchange information. In the parallel sorting application, most of the information which is shared is exchanged through the file systems. However, the message passing libraries are used to exchange some control information such *splitting keys* and to enforce some synchronization among the tasks. The following message passing calls are used in the parallel sorting application:

- MPI_Barrier
- MPI_allgather

For more information about the message passing libraries, refer to the IBM Parallel Environment documentation listed in Appendix H, "Related Publications" on page 223.

5.3.10 Results of Experiments

The results of the experiments are presented in two sets of tables and one set of charts. The first set consists of three tables (shown in Table 23 on page 171, Table 24 on page 172, and Table 25 on page 173) to represent the results of running the parallel sort application on three different file sizes respectively. The three file sizes used in this experiment are 32MB, 128MB, and 256MB. In each of these three tables, the data is grouped by the number of clients/tasks used in the parallel sort application. The parameter, which is the number of clients, takes values 1, 4, 8, and 12. For each client size up to nine file systems are used. The abbreviations used to label the file systems in all the reports and charts are listed in Table 22 on page 170.

Table 22. Abbreviations Used for the File Systems

Label	**Description**
JFS-1-disk	Journal File System with one server and one SSA disk
JFS-6-disk	Journal File System with one server and 6 SSA disks
NFS-1-disk	Network File System with one SSA disk
NFS-6-disk	Network File System with six SSA disks
GPFS-1-SRVR-1-disk	General Parallel File System with one server and one SSA disk
GPFS-1-SRVR-6-disk	General Parallel File System with one server and six SSA disks
GPFS-4-SRVR-1-disk	General Parallel File System with four servers and one SSA disk
GPFS-8-SRVR-1-disk	General Parallel File System with 8 servers and one SSA disk
GPFS-12-SRVR-1-disk	General Parallel File System with 12 servers and one SSA disk

Table 23. Results for Parallel Sort Application (File Size=32MB)

File Size=32MB

Clients	File System	Read Time(sec)	Read Bandwidth	Write Time(sec)	Write Bandwidth	Compute Time(sec)	Total Time(sec)
1	JFS-1-disk	7.4	8.7	10.3	6.2	53.6	71.3
	JFS-6-disk	10.2	6.2	2.8	22.7	53.9	67.0
	NFS-1-disk	23.7	2.7	13.9	4.6	51.8	89.4
	NFS-6-disk	19.9	3.2	9.1	7.0	52.0	80.9
	GPFS-1-SRVR-1-disk	18.9	3.4	17.0	3.8	56.6	92.5
	GPFS-1-SRVR-6-disk	19.5	3.3	16.3	3.9	54.0	89.9
	GPFS-4-SRVR-1-disk	20.3	3.1	20.5	3.1	50.70	91.5
	GPFS-8-SRVR-1-disk	20.0	3.2	20.5	3.1	50.6	91.2
	GPFS-12-SRVR-1-disk	20.9	3.1	21.6	3.0	50.6	93.0
4	JFS-1-disk	4.9	13.2	0.6	104.6	55.9	61.3
	JFS-6-disk	1.1	56.6	0.5	136.1	55.5	57.1
	NFS-1-disk	7.8	8.3	5.3	12.0	41.9	55.0
	NFS-6-disk	8.0	8.0	5.0	12.8	29.2	42.2
	GPFS-1-SRVR-1-disk	22.7	2.8	15.0	4.3	19.4	57.1
	GPFS-1-SRVR-6-disk	8.3	7.7	6.9	9.3	17.7	32.9
	GPFS-4-SRVR-1-disk	13.5	4.7	8.4	7.6	30.5	52.4
	GPFS-8-SRVR-1-disk	11.4	5.6	8.0	8.0	18.2	37.5
	GPFS-12-SRVR-1-disk	10.4	6.2	5.8	11.0	20.1	36.3
8	NFS-1-disk	6.1	10.6	6.3	10.1	30.0	42.4
	NFS-6-disk	5.3	12.1	3.3	19.3	25.7	34.3
	GPFS-1-SRVR-1-disk	19.9	3.2	9.4	6.8	22.7	52.0
	GPFS-1-SRVR-6-disk	7.9	8.1	3.6	17.8	15.2	26.8
	GPFS-4-SRVR-1-disk	10.5	6.1	6.0	10.6	23.3	39.8
	GPFS-8-SRVR-1-disk	9.1	7.1	3.2	19.8	19.8	32.1
	GPFS-12-SRVR-1-disk	7.9	8.1	3.8	17.0	25.4	37.1
12	NFS-1-disk	8.2	7.8	8.6	7.5	25.0	41.8
	NFS-6-disk	6.6	9,7	3.1	21.0	25.6	35.2
	GPFS-1-SRVR-1-disk	17.4	3.7	6.4	9.9	30.2	54.1
	GPFS-1-SRVR-6-disk	5.9	10.9	6.6	9.7	19.9	32.4
	GPFS-4-SRVR-1-disk	10.8	5.9	8.7	7.3	31.2	50.8
	GPFS-8-SRVR-1-disk	7.7	8.4	5.9	10.9	18.6	32.1
	GPFS-12-SRVR-1-disk	6.7	9.6	6.4	9,9	20.9	34.0

Table 24. Results for Parallel Sort Application (File Size=128MB)

File Size=128MB

Clients	File System	Read Time(sec)	Read Bandwidth	Write Time(sec)	Write Bandwidth	Compute Time(sec)	Total Time(sec)
1	JFS-1-disk	62.3	4.1	17.2	14.9	244.0	323.5
	JFS-6-disk	50.7	5.1	7.9	32.4	244.0	302.6
	NFS-1-disk	130.9	1.9	87.3	2.9	245.2	464.5
	NFS-6-disk	108.4	2.4	66.8	3.8	243.5	418.8
	GPFS-1-SRVR-1-disk	115.3	2.2	69.6	3.7	251.7	436.7
	GPFS-1-SRVR-6-disk	121.6	2.1	68.3	3.7	251.6	441.6
	GPFS-4-SRVR-1-disk	152.7	1.7	86.9	2.9	251.9	491.4
	GPFS-8-SRVR-1-disk	168.4	1.5	90.0	2.8	252.9	511.3
	GPFS-12-SRVR-1-disk	175.2	1.5	93.9	2.7	250.3	519.4
4	JFS-1-disk	78.6	3.3	11.9	21.4	276.0	366.5
	JFS-6-disk	78.0	3.3	10.0	25.6	264.4	352.4
	NFS-1-disk	156.8	1.6	70.8	3.6	189.1	416.7
	NFS-6-disk	100.9	2.5	27.3	9.4	175.2	303.4
	GPFS-1-SRVR-1-disk	139.8	1.8	38.1	6.7	84.3	262.2
	GPFS-1-SRVR-6-disk	50.6	5.1	30.3	8.4	81.7	162.7
	GPFS-4-SRVR-1-disk	77.8	3.3	33.1	7.7	90.4	201.3
	GPFS-8-SRVR-1-disk	63.3	4.0	29.2	8.8	84.9	177.4
	GPFS-12-SRVR-1-disk	63.3	4.0	25.5	10.1	94.0	182.7
8	NFS-1-disk	200.3	1.3	31.4	8.2	155.6	387.3
	NFS-6-disk	91.2	2.8	22.4	11.4	174.2	287.8
	GPFS-1-SRVR-1-disk	130.6	2.0	37.5	6.8	69.1	237.2
	GPFS-1-SRVR-6-disk	37.8	6.8	14.0	18.2	60.1	112.0
	GPFS-4-SRVR-1-disk	58.4	4.4	20.6	12.4	64.4	143.5
	GPFS-8-SRVR-1-disk	51.7	5.0	13.0	19.6	55.2	119.9
	GPFS-12-SRVR-1-disk	37.3	6.9	15.0	17.1	62.5	114.9
12	NFS-1-disk	148.9	1.7	38.3	6.7	195.4	382.6
	NFS-6-disk	63.2	4.1	17.3	14.8	185.4	265.9
	GPFS-1-SRVR-1-disk	113.9	2.2	39.9	6.4	68.7	222.5
	GPFS-1-SRVR-6-disk	33.3	7.7	13.4	19.2	48.6	95.3
	GPFS-4-SRVR-1-disk	50.5	5.1	17.0	15.0	77.9	145.4
	GPFS-8-SRVR-1-disk	31.1	8.2	15.8	16.2	67.4	114.3
	GPFS-12-SRVR-1-disk	30.1	8.5	19.6	13.0	47.0	96.7

Table 25. Results for Parallel Sort Application (File Size=256MB)

File Size=256MB

Clients	File System	Read Time(sec)	Read Bandwidth	Write Time(sec)	Write Bandwidth	Compute Time(sec)	Total Time(sec)
1	JFS-1-disk	113.5	4.5	40.4	12.7	630.5	784.5
	JFS-6-disk	106.2	4.8	18.7	27.3	628.5	753.6
	NFS-1-disk	272.2	1.9	214.5	2.4	631.0	1117.7
	NFS-6-disk	211.5	2.4	162.7	3.1	629.2	1003.6
	GPFS-1-SRVR-1-disk	350.2	1.5	140.4	3.6	642.7	1133.4
	GPFS-1-SRVR-6-disk	374.9	1.4	139.0	3.7	637.9	1151.9
	GPFS-4-SRVR-1-disk	472.7	1.1	179.4	2.9	646.1	1298.4
	GPFS-8-SRVR-1-disk	510.4	1.0	186.4	2.7	645.8	1342.7
	GPFS-12-SRVR-1-disk	533.8	1.0	193.5	2.6	655.2	1382.6
4	JFS-1-disk	111.0	2.4	20.2	25.3	663.8	895.1
	JFS-6-disk	141.8	3.6	10.3	49.5	666.4	818.6
	NFS-1-disk	508.0	1.0	99.8	5.1	387.2	995.0
	NFS-6-disk	265.9	1.9	64.3	8.0	395.2	725.4
	GPFS-1-SRVR-1-disk	349.3	1.5	56.7	9.0	228.9	635.0
	GPFS-1-SRVR-6-disk	139.5	3.7	33.0	15.5	221.8	394.4
	GPFS-4-SRVR-1-disk	219.0	2.3	30.4	16.8	214.1	463.5
	GPFS-8-SRVR-1-disk	173.6	3.0	40.7	12.6	220.4	434.7
	GPFS-12-SRVR-1-disk	173.1	3.0	44.0	11.6	222.1	439.1
8	NFS-1-disk	550.9	0.9	71.2	7.2	333.2	955.3
	NFS-6-disk	353.9	1.4	53.3	9.6	251.9	659.1
	GPFS-1-SRVR-1-disk	416.2	1.2	50.8	10.1	160.7	627.7
	GPFS-1-SRVR-6-disk	105.9	4.8	27.3	18.7	123.0	256.2
	GPFS-4-SRVR-1-disk	192.4	2.7	20.9	24.5	144.0	357.3
	GPFS-8-SRVR-1-disk	139.7	3.7	33.1	15.5	144.0	316.9
	GPFS-12-SRVR-1-disk	120.2	4.3	29.3	17.5	146.7	296.3
12	NFS-1-disk	588.7	0.9	75.9	6.7	273.3	938.0
	NFS-6-disk	388.3	1.3	48.0	10.7	229.9	666.2
	GPFS-1-SRVR-1-disk	402.6	1.3	37.6	13.6	163.9	604.0
	GPFS-1-SRVR-6-disk	99.9	5.1	16.3	31.5	105.7	221.9
	GPFS-4-SRVR-1-disk	177.9	2.9	28.4	18.0	125.3	331.6
	GPFS-8-SRVR-1-disk	115.4	4.4	20.2	25.3	103.4	239.0
	GPFS-12-SRVR-1-disk	103.3	5.0	25.3	20.2	129.7	258.3

The second set of tables also consists of three tables to represent the results of using three file sizes: 32MB, 128MB, and 256MB. This set is presented in Table 26 on page 174, Table 27 on page 175, and Table 28 on page 176. In each of these three tables, the performance data for a given type of file system is grouped together. Nine file systems are used for each of the client sizes.

Table 26. Results for Parallel Sort Application (File Size=32MB)

File Size=32MB

File System	Clients	Read Time(sec)	Read Bandwidth	Write Time(sec)	Write Bandwidth	Compute Time(sec)	Total Time(sec)
JFS-1-disk	**1**	7.4	8.7	10.3	6.2	53.6	71.3
	4	4.9	13.2	0.6	104.6	55.9	61.3
JFS-6-disk	**1**	10.2	6.2	2.8	22.7	53.9	67.0
	4	1.1	56.6	0.5	136.1	55.5	57.1
NFS-1-disk	**1**	23.7	2.7	13.9	4.6	51.8	89.4
	4	7.8	8.,3	5.3	12.0	41.9	55.0
	8	6.1	10.6	6.3	10.1	30.0	42.4
	12	8.2	7.8	8,6	7.5	25.0	41.8
NFS-6-disk	**1**	19.9	3.2	9.1	7.0	52.0	80.9
	4	8.0	8.0	5.0	12.8	29.2	42.2
	8	5.3	12.1	3.3	19.3	25.7	34.3
	12	6.6	9,7	3.1	21.0	25.6	35.2
GPFS-1-SRVR-1-disk	**1**	18.9	3.4	17.0	3.8	56.6	92.5
	4	22.7	2.8	15.0	4.3	19,4	57.1
	8	19.9	3.2	9.4	6.8	22.7	52.0
	12	17.4	3.7	6.4	9.9	30.2	54.1
GPFS-1-SRVR-6-disk	**1**	19.5	3.3	16.3	3.9	54.0	89.9
	4	8.3	7.7	6.9	9.3	17.7	32.9
	8	7.9	8.1	3.6	17.8	15.2	26.8
	12	5.9	10.9	6.6	9.7	19.9	32.4
GPFS-4-SRVR-1-disk	**1**	20.3	3.1	20.5	3.1	50.7	91.5
	4	13.5	4.7	8.4	7.6	30.5	52.4
	8	10.5	6.1	6.0	10.6	23.3	39.8
	12	10.8	5.9	8.7	7.3	31.2	50.8
GPFS-8-SRVR-1-disk	**1**	20.0	3.2	20.5	3.1	50.6	91.2
	4	11.4	5.6	8.0	8.0	18.2	37.5
	8	9.1	7.1	3.2	19.8	19.8	32.1
	12	7.7	8.4	5.9	10.9	18.6	32.1
GPFS-12-SRVR-1-disk	**1**	20.9	3.1	21.6	3.0	50.6	93.0
	4	10.0	6.2	5.8	11.0	20.1	36.3
	8	7.9	8.1	3.8	17.0	25.4	37.1
	12	6.7	9.6	6.4	9.9	20.9	34.0

Table 27. Results for Parallel Sort Application (File Size=128MB)

File Size=128MB

File System	Clients	Read Time(sec)	Read Bandwidth	Write Time(sec)	Write Bandwidth	Compute Time(sec)	Total Time(sec)
JFS-1-disk	**1**	62.3	41.1	17.2	14.9	244.0	323.5
	4	78.6	3.3	11.9	21.4	276.0	366.6
JFS-6-disk	**1**	50.7	5.1	7.9	32.4	244.0	302.6
	4	78.0	3.3	10.0	25.6	264.4	352.4
NFS-1-disk	**1**	131.9	1.9	87.3	2.9	245.2	464.5
	4	156.8	1.6	70.8	3.6	189.1	416.7
	8	200.3	1.3	31.4	8.2	155.6	387.3
	12	148.9	1.7	38.3	6.7	195.4	382.6
NFS-6-disk	**1**	108.4	2.4	66.8	3.8	243.5	418.8
	4	100.9	2.5	27.3	9.4	175.2	303.4
	8	91.2	2.8	22.4	11.4	174.2	287.8
	12	63.2	4.1	17.3	14.8	185.4	265.9
GPFS-1-SRVR-1-disk	**1**	115.3	2.2	69.6	3.7	251.7	436.7
	4	139.8	1.8	38.1	6.7	84.3	262.2
	8	130.6	2.0	37.5	6.8	69.1	237.2
	12	113.9	2.2	39.9	6.4	58.7	222.5
GPFS-1-SRVR-6-disk	**1**	121.6	2.1	68.3	3.7	251.6	441.6
	4	50.6	5.1	30.3	8.4	81.7	162.7
	8	37.8	6.8	14.0	18.2	60.1	112.0
	12	33.3	7.7	13.4	19.2	48.6	95.3
GPFS-4-SRVR-1-disk	**1**	152.7	1.7	86.9	2.9	251.9	491.4
	4	77.8	3.3	33.1	7.7	90.4	201.3
	8	58.4	4.4	20.6	12.4	64.4	143.5
	12	50.5	5.1	17.0	15.0	77.9	145.4
GPFS-8-SRVR-1-disk	**1**	168.4	1.5	90.0	2.8	252.9	511.3
	4	63.3	4.0	29.2	8.8	84.9	177.4
	8	51.7	5.0	13.0	19.6	55.2	119.9
	12	31.1	8.2	15.8	16.2	67.4	114.3
GPFS-12-SRVR-1-disk	**1**	175.2	1,5	93.9	2.7	250.3	519.4
	4	63.3	4.0	25.5	10.1	94.0	182.7
	8	37.3	6.9	15.0	17.1	62.5	114.9
	12	30.1	8.5	19.6	13.0	47.0	96.7

Table 28. Results for Parallel Sort Application (File Size=256MB)

File Size=256MB

File System	Clients	Read Time(sec)	Read Bandwidth	Write Time(sec)	Write Bandwidth	Compute Time(sec)	Total Time(sec)
JFS-1-disk	**1**	113.5	4.5	40.4	12.7	630.5	784.5
	4	211.0	2.4	20.2	25.3	663.8	895.1
JFS-6-disk	**1**	106.2	4.8	18.7	27.3	628.5	753.6
	4	141.8	3.6	10.3	49.5	666.4	818.6
NFS-1-disk	**1**	272.2	1.9	214.5	2.4	631.0	1117.9
	4	508.0	1.0	99.8	5.1	387.2	995.0
	8	550.9	0.9	71.2	7.2	333.2	955.3
	12	588.7	0.9	75.9	6.7	273.3	938.0
NFS-6-disk	**1**	211.5	2.4	162.7	3.1	629.2	1003.6
	4	265.9	1.9	64.3	8.0	395.2	725.5
	8	353.9	1.4	53.3	9.6	251.9	659.1
	12	388.3	1.3	48.3	10.7	229.9	666.2
GPFS-1-SRVR-1-disk	**1**	350.2	1.5	140.4	3.6	642.7	1133.4
	4	349.3	1.5	56.7	9.0	228.9	635.0
	8	416.2	1.2	50.8	10.1	160.7	627.7
	12	402.6	1.3	37.6	13.6	163.9	604.0
GPFS-1-SRVR-6-disk	**1**	374.9	1.4	139.0	3.7	637.9	1151.9
	4	139.5	3.7	33.0	15.5	221.8	394.4
	8	105.9	4.8	27.3	18.7	123.0	256.2
	12	99.9	5.1	16.3	31.5	105.7	221.9
GPFS-4-SRVR-1-disk	**1**	472.7	1.1	179.4	2.9	646.1	1298.4
	4	219.0	2.3	30.4	16.8	214.1	463.5
	8	192.4	2.7	20.9	24.5	144.0	357.3
	12	177.9	2.9	28.4	18.0	125.3	331.6
GPFS-8-SRVR-1-disk	**1**	510.4	1.0	186.4	2.7	645.8	1342.7
	4	173.6	3.0	40.7	12.6	220.4	434.7
	8	139.7	3.7	33.1	15.5	144.0	316.9
	12	115.4	4.4	20.2	25.3	103.4	239.0
GPFS-12-SRVR-1-disk	**1**	533.8	1.0	193.5	2.6	655.2	1382.6
	4	173.1	3.0	44.0	11.6	222.1	439.1
	8	120.2	4.3	29.3	17.5	146.7	296.3
	12	103.3	5.0	25.3	20.2	129.7	258.3

The data in tables Table 26 on page 174, Table 27 on page 175, Table 28 is also shown in a graph form in the following figures:

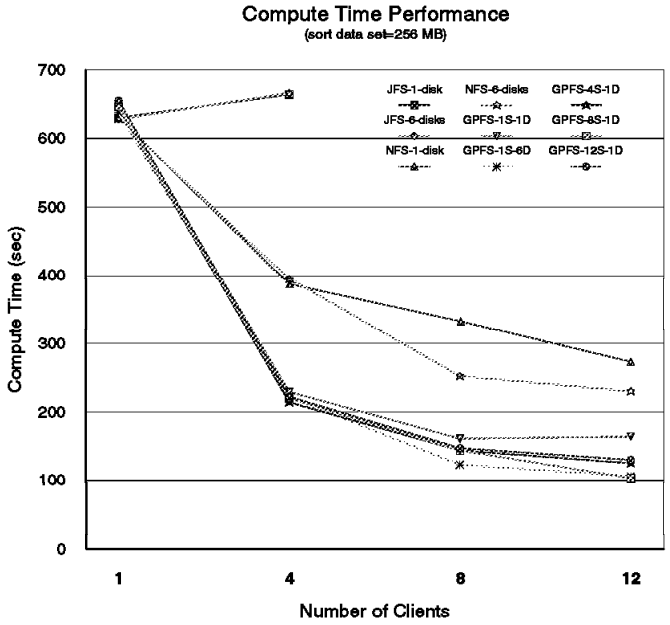


Figure 40. Compute Time Performance in the Sort Application

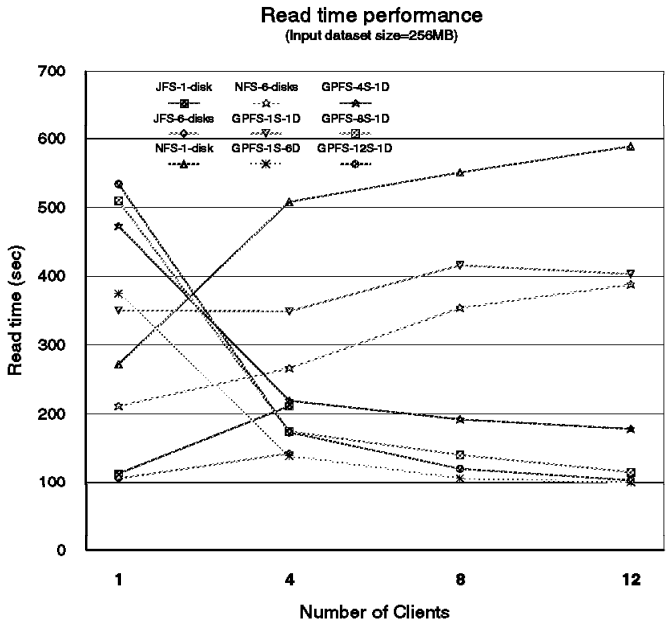


Figure 41. Read Time Performance in the Sort Application

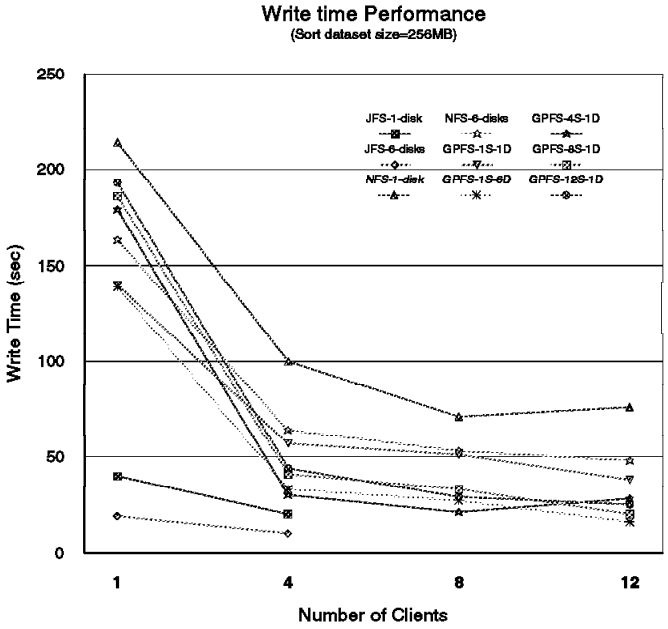


Figure 42. Write Time Performance in the Sort Application

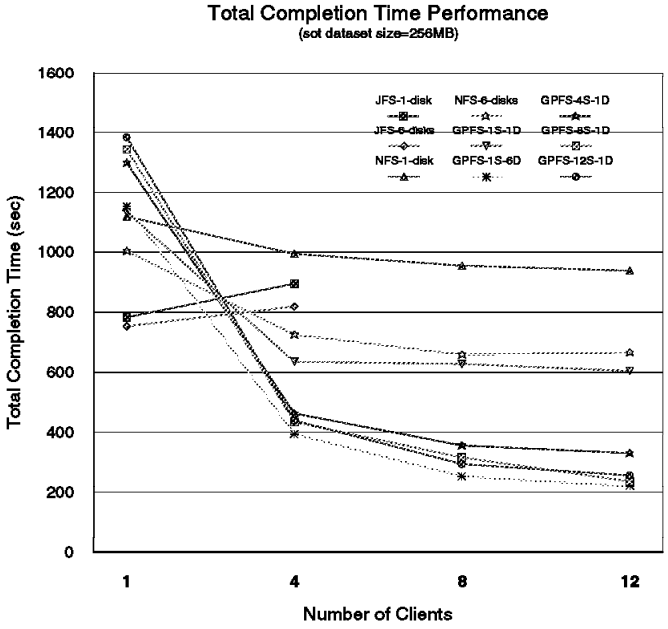


Figure 43. Total Time Performance in the Sort Application

5.3.10.1 Compute Time Performance

The entry under the heading *Compute time* shown in all the tables and the charts indicates the non-I/O time spent by each task to do work related to sorting. In the case of a single task this measure quantifies the total amount of work an application has to do outside of I/O to do

sorting. When multiple tasks/clients are used, this measure indicates the elapsed time to complete the non-I/O work by the parallel application. For example, in order to achieve linear speed-up, if a sorting application with one client takes 600 seconds to do sorting, a 12-client application should take 50 seconds to complete the job. In practice, linear speed-up is hard to achieve due to synchronization and replication overhead. For example, as shown in Table 23 on page 171 and Figure 40 on page 177 a single client sorts a GPFS file of size 256MB residing on 8 servers in 645 seconds. A parallel application with 8 clients sorts the same file in 220 seconds, achieving a speed-up of 3 instead a linear speed-up of 8. JFS configurations are tested with 1-client and 4-client configurations and do not show any speed-up since all the clients share the same processor.

5.3.10.2 Read Time Performance

The entries under the heading *read time* represent the time that the application has to wait after issuing an *fread* call to get the data from the external devices. In the current implementation the application is blocked until the data is available in the application defined data areas. It would be an interesting extension of the current work to investigate potential performance gains that may be realized by hiding the delay incurred to get the data by letting the application perform other useful work.

The following observations can be made by examining the results recorded for the *read time* measure:

1. Single application task/client scenarios
   - Both JFS configurations perform significantly better than any other file system configurations.
   - More complex configurations resulted in worse *read time* performance.
   - Multi-disk configurations, as expected, are always better than their single disk counterparts.
2. Multiple application task/client scenarios
   - All GPFS configurations with multiple Virtual Shared Disk servers including the case of single physical processor serving multiple Virtual Shared Disk servers, showed dramatic reduction in read time.
   - Among the GPFS configurations, the configuration with one server and 6 SSA disks turned out to be a strong contender for a small number of application clients and showed only slightly worse performance than its multinode counterparts when larger number of clients are used.
   - The experiments conducted with 12 application clients showed that *NFS-1-disk* results in the worst read time performance, and the most complex and expensive GPFS configuration results in the best read time performance.

5.3.10.3 Write Time Performance

The entries under the heading *write time* represent the time application has to wait after issuing an *fwrite* call to get the data from the application buffer into the system buffers. The return from the *fwrite* call does not mean that the data is actually written to the disk. This feature of the AIX system improves the performance of the application by hiding the delay involved in writing the data behind any unrelated and useful processing the application can do while the data transfer is in progress. Since this is the common mode of write operation this feature of the AIX Operating System is not turned off in this study. It would be an interesting extension of this work to observe the write performance with this feature turned off.

The following observations can be made by examining the results recorded for the *write time* measure:

1. Single application task/client scenarios
    - Both JFS configurations performed significantly better than any other file system configurations.
    - NFS with one disk configuration performed significantly worse than any other file system configurations.
    - Multidisk configurations resulted in better performance across all file systems using single processor as server than the same systems with a single disk.
    - The performance of GPFS configurations exhibited an inverse relationship to their cost and complexity. For example, *GPFS-12-SRVR-1-Disk* resulted in significantly worse performance than *GPFS-1-SRVR-1-Disk*.
2. Multiple application task/client scenarios
    - All file system configurations showed dramatic reduction in write time.
    - More complex GPFS file system configurations resulted in more reduction in write time than their less complex counterparts for smaller number of clients. For a larger number of clients, more complex GPFS configurations seem to perform better.
    - It is interesting to see that the performance of *GPFS-1-SRVR-7-Disk* configuration is very close to that of more complex configurations.

5.3.10.4 Total Completion Time Performance

The total completion time is the summation of the I/O time and compute time. All configurations showed improvement in this measure as the number of clients increase. This can be explained from the fact that the total amount non-I/O work is very large compared to the I/O time and all the file systems showed improvement in the compute time as more clients are added. This improvement masked the significant degradation in the read time in the case for NFS file systems. For complex GPFS configurations the variation in the total completion time statistic is not very significant. For example, the completion times for

the *GPFS-12-SRVR-1-Disk* and *GPFS-8-SRVR-1-Disk* configurations are 258 seconds and 238 seconds, respectively. Further investigations with a larger client set are needed to see if more complex configurations justify the additional cost and complexity.

5.3.11 Conclusions

The following conclusions can be drawn from the results of this experiment:

1. For the single-node configuration, JFS clearly outperforms other file systems.
2. NFS seems to be a contender for distributed systems with a small number of clients accessing the file system. As the number of clients increases, the performance of NFS degrades rapidly and it is not an alternative for file systems which serve large number of remote clients.
3. GPFS configurations based on a single node with a single disk seems to perform significantly better than the two NFS configurations (the one with a single disk and the one with multiple disks).
4. It is interesting to see that the GPFS configuration based on a single processor with multiple disks is a strong contender for most of the scenarios used in this study.
5. For systems with a small number of clients, larger GPFS configurations do not seem to be strong contenders.
6. The benefits of larger GPFS configurations seem to surface quickly even as very small number of clients access the file system. This is due to the fact that requests from the clients can be processed in parallel by multiple servers in the GPFS configuration.
7. Even for the small number of clients used in the parallel sort application, the GPFS configuration with 12 servers exhibited better performance than smaller configurations.

Appendix A. GPFS and Standard AIX Commands

This appendix provides a list of equivalences between AIX and GPFS file system commands.

A.1 File Commands

Table 29 lists GPFS and AIX files manipulation commands.

Table 29 (Page 1 of 3). List of AIX File Commands

Command	Action	On GPFS File	Comments
backup -i	Backs up files by name	OK	
backup	Back up file systems by inode	NO	Message backup: 0511-028 Not a native file system.
cat	Concatenates and displays	OK	
chgrp	Changes the group ownership	OK	
chmod	Changes the permission modes	OK	
chown	Changes the ownership	OK	
cksum	Checks checksum and byte count	OK	
copy,cp	Copies	OK	From,to and inside GPFS file system.
cpio	Archives(-o), Restores(-i)	OK	From,to and inside GPFS file system.
dd	Converts and copy	OK	Do not try to dd the complete file system (/dev/fs1) to another type of file system because of differences in the superblock, metadata, and data structures.
del	Deletes with confirmation	OK	

Table 29 (Page 2 of 3). List of AIX File Commands

Command	Action	On GPFS File	Comments
file	Determines file type	OK	
fileplace	Determine file placement in a fs,lv,pv with space efficiency and fragmentation information	NO	Message: could not identify device which contains 'file_name'.
fuser	Gives information about processes using specific files	NO	Does not give any information on processes using files inside a GPFS file system (lsof does not give more information).
ln	Links files inside a file system	OK	
ls	Lists files included in a directory with/without statistics	OK	
ln -s	Links files between different file systems	OK	From,to and inside GPFS file system.
mv	Moves files	OK	From,to and inside GPFS file system.
pax	Archives(-w), Restores(-r)	OK	From,to and inside GPFS file system.
restore -xd	Restores by name (a backup by name archive)	OK	From,to and inside GPFS file system.
restore	Restores by inode (a backup by inode archive)	OK	To a GPFS file system.
rm	Removes files and directories	OK	
split	Splits a file into pieces	OK	
sum	Displays the checksum and block count of a file	OK	

Table 29 (Page 3 of 3). List of AIX File Commands

Command	Action	On GPFS File	Comments
sum	Displays the checksum and block count of a file	OK	
tar	Archives(-c), Restores(-x) files	OK	
touch	Updates access and modification time of a file	OK	

A.2 File System Commands

Table 30 lists GPFS and AIX file system commands.

Table 30 (Page 1 of 3). List of AIX File System Commands

Command	Action	On GPFS file	Comments
chfs	Changes attributes of a file system	Use `mmchfs`	Do not use this command with GPFS file systems because mmfs SDR files are not updated by this command. Use `mmchfs` instead.
crfs	Adds a file system	Use `mmcrfs`	Do not use this command with GPFS file systems because mmfs SDR files are not updated by this command. Use `mmcrfs`, instead.
defragfs	Reorganizes data inside a file system	NO	Message defragfs: 0506-408 failure in opening device /dev/fs1.
df	Reports information about space on file systems	OK, but...	Updates on the file systems made by other VSD nodes are not refreshed immediately on a VSD nodes. Use `mmdf fs_name` instead. `mmdf` can be used on an unmounted file system.
dumpfs	Dumps file system information	NO	Message: 0506-013 cannot read superblock on /dev/fs1. There is no equivalent command for GPFS file systems.

Table 30 (Page 2 of 3). List of AIX File System Commands

Command	Action	On GPFS file	Comments
filemon	Gives usage information of fs lv pv vg	OK, but...	Does not give any file system information on GPFS file system, only gives informations on underlying lvs.
ff	Lists file names and statistics for a file system	NO	Message: Filesystem Helper: 0506-0519 device open failed. Use du instead.
fsck	Checks file system consistency and does repairs	Use `mmfsck`	Message: fsck: 0506-035 /dev/fs1 is not a known file system. Use `mmfsck fs_name` instead. mmfsck cannot be used on a mounted file system.
fsdb	Debugs file systems	Use `mmlsfs`	Messages: WARNING and FATAL_ERROR reading superblocks. Use `mmlsfs fs_name` instead. (Command fsdb is only adapted to jfs.)
istat	Examines i-node numbers	OK, but...	Only istat f_name. When entering istat inode_number device Message: istat: 0506-013 cannot read superblock on /dev/fs1.
lsfs	Displays the characteristics of file systems	OK	
mount	Makes a file system available for use	OK	
rmfs	Removes a file system and underlying logical volume	Use `mmdelfs`	Never use this command. Use `mmdelfs` instead.

Table 30 (Page 3 of 3). List of AIX File System Commands

Command	Action	On GPFS file	Comments
rrestore	Restores a file created with backup command from a remote location	OK	
xlvm	vsm application enabling GUI storage management	NO	No template exists yet for taking GPFS file system into account.

Appendix B. GPFS Maximum File Size and Related Parameters

Table 31 (Page 1 of 2). GPFS Maximum File Size and Related Parameters

(B) Block Size	(I) Indirect Size	(i) I-Node Size	(M,D) Maximum Replication	Approx. Maximum File Size	Approx. Maximum File System Size
16KB	512	512	1	85MB	58GB
			2	21MB	14GB
16KB	1KB	512	1	178MB	115.7GB
			2	44MB	28.9GB
16KB	4KB	512	1	735MB	478.4GB
			2	184MB	119.6GB
16KB	16KB	512	1	2.8GB	†1TB
			2	741MB	482.3GB
64KB	4KB	512	1	2.8GB	†1TB
			2	735MB	†1TB
64KB	4KB	1KB	1	6.3GB	†1TB
			2	1.6GB	†1TB
64KB	4KB	2KB	1	13.3GB	†1TB
			2	3.3GB	†1TB
64KB	16KB	512	1	11.3GB	†1TB
			2	2.8GB	†1TB
64KB	16KB	1KB	1	25.5GB	†1TB
			2	6.4GB	†1TB
64KB	16KB	2KB	1	53.8GB	†1TB
			2	13.5GB	†1TB
64KB	32KB	512	1	22.6GB	†1TB
			2	5.7GB	†1TB
64KB	32KB	1KB	1	51GB	†1TB
			2	12.8GB	†1TB
64KB	32KB	2KB	1	107.8GB	†1TB
			2	26.9GB	†1TB
256KB	16KB	512	1	45.2GB	†1TB
			2	11.3GB	†1TB
256KB	16KB	4KB	1	442.3GB	†1TB
			2	110.6GB	†1TB
256KB	32KB	512	1	90.5GB	†1TB
			2	22.6GB	†1TB

Table 31 (Page 2 of 2). GPFS Maximum File Size and Related Parameters

(B) Block Size	(I) Indirect Size	(i) I-Node Size	(M,D) Maximum Replication	Approx. Maximum File Size	Approx. Maximum File System Size
256KB	32KB	4KB	1	885.9GB	†1TB
			2	221.5GB	†1TB

Note: † When a combination of parameters yields to a maximum file system size greater than one terabyte (1 TB), consider one terabyte (1 TB) as the maximum file system size.

Appendix C. The GPFS Configuration File (mmsdrcfg1)

This file is stored in the SDR and it is partition-sensitive. It provides a set of parameters taken by the GPFS daemon at startup time. The following is a description of each one of the variables set in this file.

Note

GPFS is absolutely dependent on this file. No modification should be made to this file without a complete understanding of what effect that change will have in the GPFS environment.

```
# Sample configuration file

## Numbers may end with a single letter:
##   k or K meaning 1024
##   m or M meaning 1048576 (1024*1024)

########  Memory / Shared Segment Configuration  ########

## The pagepool and malloc area will always reside in pinned memory.
pagepool 20M
mallocsize 4M

## Maximum number of files to cache. The default value is 1000. This number
## will be reduced to a number that takes no more than about 50% of mallocsize.
## If the number of concurrently open files is bigger, then the number of cached
## files will exceed this value.
maxFilesToCache 200


########  Multi-node configuration  ########

# The following parameters are used to configure Tiger Shark to allow
# multiple file system nodes to simultaneously access files stored on
# the same set of shared disks, for example on SP/2 using virtual shared
# disks (VSD).

# Change this to yes to enable multi-node file systems
multinode yes

## Inter-node communication configuration
tscTcpPort     6667           # Tcp port number

##   Setting tscWorkerPool to zero disables inter-node communication.
tscWorkerPool  10             # Number of worker threads
```

```
##   If wait4RVSD is yes, then before becomes ready for service, the
##   mmfsd daemon waits for RVSD to become active.
##   This flag is meaningfull only in the presence of Group Services.
wait4RVSD no

##   The default Group Service names are: MmfsGroup and MmfsRecGroup
##   Use these parameters to override the default group names.
##   This is necessary when more than one group of nodes running
##   separate MMFS clusters in the same SP partition.
##   Names may be up to 16 characters long.
# group newmmfsgroupname
# recgroup newmmfsrecgroupname

##   To use a file other than /etc/cluster.nodes as a node list.
# nodelist new-file-name

##   To specify a file other than /var/mmfs/etc/cluster.preferences
##   as a node list of preferred administration nodes.
# nodeprefs new-file-name

# The next three parameters control the maximum degree of multiprogramming
# within the mmfsd daemon.  The value of the 'prefetchThreads' parameter
# controls the maximum possible number of threads dedicated to prefetching
# data for files that are read sequentially.  The actual degree of
# parallelism for prefetching is determined dynamically by the daemon.
# The 'worker1Threads' parameter controls the maximum number of threads
# that are used to handle sequential writebehind, as well as some other
# operations associated with data access.  This value should be 150% of
# the value of 'prefetchThreads'.  The 'worker2Threads' parameter controls
# the maximum number of threads that are used to handle miscellaneous
# operations, mainly those involving directories (open, stat, mkdir, etc.).
# This parameter should be set to 50% of the value for the 'prefetchThreads'
# parameter.  The minimum values for all 3 parameters is 1, and the maximums
# are 48, 72, and 24, respectively.
prefetchThreads 24
worker1Threads 36
worker2Threads 12

# Timeout to assign a new stripe group manager after the old one
# has resigned or failed. Defualt value is 600 seconds.
# If a new stripe manager that functions cannot be assigned within
# this timeout period, the config manager will broadcast a message
# saying that there is no manager for this sg, and force all nodes to
# panic this SG. Timer resets when a stripe group manager notifies the
# config manager that it actually has mounted the SG.
takeovertimeout 600
```

```
########  Problem determination Configuration  ########

# Tracing of individual classes of events/operations can be activated by
# adding "trace <trace-class> <trace level>" lines below.
trace all 0

## The 'crashdump' keyword controls how much data from mmfs is
## included in a system dump, should AIX crash.  A value of 0 disables
## putting any mmfs data in system dumps.  A value of 1 dumps
## the prefix and the malloc area of the shared segment.  Higher values
## are reserved for future use to dump additional data areas.
crashdump 0

## The 'dataStructDump' keyword controls whether mmfs will produce a
## formatted dump of its internal data structures into a file named
## internaldump.<daemon pid>.signal whenever it aborts.
## The following entry can either be a directory name in which the file
## will reside, or otherwise a boolean value. When given a positive
## boolean value the directory defaults to /tmp/mmfs.
dataStructureDump yes

## The 'dataStructDumpOnSGPanic' keyword controls whether mmfs will
## produce a formatted dump of its internal data structures into
## a file named internaldump.<daemon pid>.<sgname> whenever a stripe
## group is panicked.
## The following entry can either be a directory name in which the file
## will reside, or otherwise a boolean value. When given a positive
## boolean value the directory defaults to /tmp/mmfs.
## Note that on real-time systems, one SG panic dumping may cause glitches
## on other SGs, and therefore is not recommended for production.
dataStructureDumpOnSGPanic yes

## The 'assertOnStructureError' keyword controls whether mmfs will
## assert and make a dump when an internal file structure error is detected.
## The structure error is always logged to the AIX error log.
assertOnStructureError no


########  Real-Time Configuration  ########

## Performance of IBM Multimedia Server for AIX can be improved by pinning the
## master process, and giving it a priority higher than the time-slicer.

# The 'priority' keyword sets the Unix dispatching priority of the daemon
# process to the given value.
# Recommendations:
#   Real-time systems should set this value to 4 (range 1-15)
```

```
#   Non-real-time systems should set this value to 40 (range 16-64)
priority 40

## To enable pinning change the value of the 'pindaemon' parameter to yes.
pindaemon  no

## The 'pinmaster' keyword allows tuning the size of the stack and heap (data)
## area of the master process that need to be pinned if pindaemon is enabled.
## If mmfsd fails with an assertion in cortn.C when pindaemon is enabled,
## try incrementing the value for 'data:'.
pinmaster stack:256K data:4096K

## The 'stickypool' keyword indicates how much of pagepool should be
## available for block 0 of files with the sticky-bit on.
## If the value is 0, this feature is disabled.
stickypool 0


######## Node Override Configuration  ########

##   In a multi-node configuration, it may be desirable to configure some
##   nodes differently than others.  This can be accomplished by placing
##   separate, potentially different, copies of the mmfs.cfg file on each
##   node.  However, since maintaining separate copies of the configuration
##   file on each node will likely be more difficult and error prone,
##   the same effect can be achieved via node overrides wherein a single
##   mmfs.cfg file is replicated on every node.
##
##   A node override is introduced by a line containing a node name or list
##   of node names in square brackets.  The node names used are those
##   listed in the /etc/cluster.nodes file.  All parameter specifications
##   that follow will apply only to the listed nodes.  A "·common“" line
##   ends a section of node overrides.  For example the following fragment:
##
##     pagepool 30M
##
##     [tiger5,tiger6]
##     pagepool 10M
##
##     [tiger9]
##     pagepool 64M
##
##     [common]
##     stickypool 2M
##
##   configures the page pool on most nodes as 30 megabytes.  However,
##   on tiger5 and tiger6 the page pool is configured with a smaller
##   page pool of 10 megabytes.  On tiger9 the page pool is configured
```

```
##   with a larger page pool of 64 megabytes.  Lines after the "·common"" line
##   again apply to all nodes ie every node will have a sticky pool of 2M.
```

C.1 How to Change the Default Values

To change the values in the GPFS configuration files, use the following steps:

1. Retrieve the file from the SDR.

   Make sure you are working with the right partition:

   ```
   [spcw:/tmp]# SDRRetrieveFile mmsdrcfg1 /tmp/mmsdrcfg1
   ```

2. Edit the file and change the appropriate variable(s).
3. Restore the file into the SDR.

   ```
   [spcw:/tmp]# SDRReplaceFile /tmp/mmsdrcfg1 mmsdrcfg1
   ```

4. Restart the GPFS daemon on the nodes.

Appendix D. SSA Configuration

With the maymap tool, you can generate a draw of the SSA loops of the SSA nodes. In the same draw you have a disk identifier which will help you to know which disks belong to which loop.

You will need this information when creating the VSDs for the GPFS file system in order to define the appropriate VSD server primary node for each VSD.

IBM employees can obtain a maymap package from:

TOOLS SENDTO YKTVMV TOOLS AIXTOOLS GET MAYMAP PACKAGE

The maymap package will be sent to your virtual reader as a compressed tar file.

A maymap output example is shown on the following page.

Following is an example of maymap output:

```
*******************************************************************************
MAYMAP Version v1.98f 21-Mar-97 16:10 : Colonial SSA Network Map Utility
Written by Lee Sanders - Copyright (C)IBM 1994/1995 - IBM Internal Use Only
*******************************************************************************
maymap: No IPN Node Number for ssa0 obtained from ODM.
maymap: Getting default IPN Node Number
maymap: Default IPN Node Number = 0x80
maymap: Number was greater than 100. Using default
*******************************************************************************
Colonial Card: ssa0 [Node 0x80] [Bus: 0] [Slot: 5]
Time Out Period set to 120
*******************************************************************************
Device ssa0 [Node 0x80] - 8 DriverPhysical resources reported
Allicat/SSA  = 0664 S1H ... Starfire/Scorfire SSA = DFHC (Model Varies)
                            Scorpion/SSA          = DCHC (Model Varies)
*******************************************************************************
.-------.   .--------.   .RESERVED.   .--------.   .RESERVED.
|sp21n11|   |Mystery |   |DFHC C4B|   |DFHC C4B|   |DFHC C4B|
|Node 80|---|?? ?? ??|---|AC7E5D69|---|AC9E14EA|---|AC9DD09B|---.
|Port A1|   |SSA Node|   |RAMSC081|   |RAMSC081|   |RAMSC081|   |
.-------.   .--------.   .--------.   .--------.   .--------.   |
        .-------------------------------------------------------.
        |  .RESERVED.
        |  |DFHC C4B|
        .--|AC7C546A|---.
           |RAMSC081|   |
           .--------.   |
.-------.               |
|sp21n11|               |
|Node 80|---------------.
|Port A2|
.-------.
.-------.   .--------.   .--------.   .--------.   .--------.
|sp21n11|   |DFHC C4B|   |DFHC C4B|   |DFHC C4B|   |DFHC C4B|
|Node 80|---|AC9DCAFB|---|AC9E2A48|---|AC7E5DDB|---|AC9E17CE|---.
|Port B1|   |RAMSC081|   |RAMSC081|   |RAMSC081|   |RAMSC081|   |
.-------.   .--------.   .--------.   .--------.   .--------.   |
        .-------------------------------------------------------.
        |  .--------.
        |  |Mystery |
        .--|?? ?? ??|---.
           |SSA Node|   |
           .--------.   |
.-------.               |
|sp21n11|               |
|Node 80|---------------.
|Port B2|
.-------.
```

Appendix E. Miscellaneous NotesBench Information

Further details about NotesBench can be found at URL:

http://www.notesbench.org

Mail and Shared DatabaseTest Script

```
 * Pause a random interval so multiple processes are staggered
 * well
 pause 0-3min
 beginloop
 * Open mail database
 changeto [MailServer]!!mail[#].nsf mail4.ntf
 * Open the current view
 open
 entries 1 20
 * Wait 5-10 seconds to peruse the view
 pause 5-10sec
 * Open 5 documents in the mail file and read each for 10-20
 * seconds
 navigate 1
 pause 10-20sec
 navigate 1
 pause 10-20sec
 navigate 1
 pause 10-20sec
 navigate 1
 pause 10-20sec
 navigate 1
 pause 10-20sec
 * Categorize 2 of the documents
 getall
 stamp 2
 * Compose 2 new mail memos/replies (taking 1-2 minutes to
 * write them)
 pause 1-2min
 sendmessage [NormalMessageSize][NumMessageRecipients][NthIteration]
 add 1
 pause 1-2min
 add 1
 * Stare at the view for 5-10 seconds and mark a few
 * documents deleted
 pause 5-10sec
 * Delete the documents which are marked for deletion (as
 * part of closing)
 getall
```

```
delete 2
* Close the view
close
* Pause at the desktop for 4-8 minutes while having a
* meeting in office
pause 4-8 min
*
* Open a discussion database
changeto [MailServer]!!testdisc.nsf discuss4.ntf
* Open the current view
open
entries 1 20
* Wait 5-10 seconds to peruse the view
pause 5-10sec
* Page down the view 2 times spending 3-10 seconds to read
* each window
entries 21 20
pause 3-10sec
entries 41 20
pause 3-10sec
* Set the unread list to a randomly selected 30 documents
unread 30
* Open next 3 unread documents and read each for 10-30
* seconds
navigate 1 next_unread
pause 10-30sec
navigate 1 next_unread
pause 10-30sec
navigate 1 next_unread
pause 10-30sec
* Close the view
close
* Pause at the desktop for 4-8 minutes while having a
* meeting in office
pause 4-8min
*
* Repeat entire sequence all over again (go back to
* beginloop statement)
rewind
```

Child notes.ini file

```
DEBUG_OUTFILE=/log/res1
ResultsDirectory=/log
NodeName=Child1
Tmp=/tmp
NthIteration=6
NormalMessageSize=1000
NumMailNotesPerUser=100
NumMessageRecipients=3
NumSharedNotes=100
```

Parent notes.ini file

```
NodeName=Parent
ResultsDirectory=/log
DEBUG_OUTFILE=/log/parent.dmp
Tmp=/tmp
NumClients1=100
NumClients2=100
NumClients3=100
NumClients4=100
NumClients5=100
NumClients6=100
NumClients7=100
NumClients8=50
NthIteration=6
NormalMessageSize=1000
NumMailNotesPerUser=100
NumMessageRecipients=3
NumSharedNotes=100
```

Output example of the NotesBench command: notesnum maildb 8

```
09/19/97 11:06:59 PM (Notes Version: 145 - UNIX) /opt/lotus/notes/latest/ibmpow/
notesnum maildb 8
Reading results file - /log/res0 ...

Min Start Time = 09/19/97 05:46:02 PM    Max Stop Time = 09/19/97 10:57:01 PM
Total Test Errors = 0
Total Test Time = 18660 sec

Calibration: Users =  100    NotesMark =   235    Response Time = 644 msec (09/
19/97 05:46:00 PM to 09/19/97 10:57:00 PM)


Reading results file - /log/res1 ...

Reading results file - /log/res2 ...

Reading results file - /log/res3 ...

Reading results file - /log/res4 ...

Reading results file - /log/res5 ...

Reading results file - /log/res6 ...

Reading results file - /log/res7 ...

Reading results file - /log/res8 ...

Min Start Time = 09/19/97 05:46:02 PM    Max Stop Time = 09/19/97 11:00:48 PM
Total Test Errors = 0
Total Test Time = 18840 sec

Test Run:    Users =  750    NotesMark =  1749    Response Time = 695 msec (09/
19/97 06:32:00 PM to 09/19/97 10:57:00 PM)


The test run/calibration response time ratio is 1.08.


Warning: The calibration file /log/res0 does not exist.
       : /log/res1 will be used as a calibration file
         (i.e., res1 will be copied to res0)

Note   : Successfully copied res1 to res0...continuing with roll up


Note:  Poller Data not collected.  Either the Poller was not run, or the
       Poller Data file '/log\pout' was deleted
```

The following section provides summaries of monitoring data in different scenarios.

Local JFS configuration with high node

Following are summaries of system monitoring during various tests when running 750 users (taken from peak time):

- iostat

```
iostat 5
--------
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        533.4               8.4      4.1       42.5      45.1

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn

hdisk16         10.6     108.8      15.0        256       288
hdisk17          9.0      97.6      12.4        296       192
hdisk18          2.6      36.8       5.6        168        16
hdisk20         32.0     225.5      33.8        420       708
hdisk22          3.0      44.0       7.0        148        72
hdisk23          8.0      55.2      10.6        172       104
```

- vmstat

```
vmstat 5
--------
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 1  1 67914   124   0   0   2  33   88   0 755 1481 251  8  3 58 31
 0  2 67914   122   0   2   8 113  248   0 837 2694 451 14  5 38 42
 1  1 67914   124   0   1   3  51  160   0 801 2414 500 12  5 52 31
 1  1 67914   142   0   1   3  59  146   0 783 1745 299  9  5 54 32
 0  1 67914   143   0   1   4  69  170   0 798 1874 296  9  4 43 43
 1  1 67914   135   0   0   9  84  268   0 826 2576 477 14  5 45 36
 1  1 67914   123   0   1   6  47  189   0 849 2306 420 12  5 37 46
 1  3 67914   148   0   6  41 130  532   0 994 2778 680 14  7  4 75
 1  2 67914   125   0   1   3  77  238   0 922 2314 560 12  6  3 80
 0  1 67914   124   0   0   3  48  125   0 816 2174 436 11  4 45 40
```

- sar

```
sar -P ALL 5
------------
AIX sp21n01 2 4 00091241A400    09/20/97

12:21:09 cpu    %usr    %sys    %wio   %idle

15:37:16  0       14       5      30      52
          1       12       6      29      53
          2        6       4      31      59
          3       10       2      31      58
          4        7       3      32      58
          5        5       1      32      62
          -        9       3      31      57
15:37:21  0       12       6      24      58
          1       10       5      24      61
          2        8       5      25      62
          3       15       4      23      58
          4        4       2      29      65
          5        6       1      28      65
          -        9       4      25      61
```

GPFS configuration with high node

Following are summaries of system monitoring during various tests when running 750 users (taken from peak time):

- iostat

```
iostat
------

VSD server sp21n11
------------------
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        171.4               0.6     11.4       20.4      67.6

(We only show you the three disks/VSDs for which this node is
VSD primary server)

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk6          35.8     614.6      25.0       1447      1626
hdisk7          38.8     769.2      28.6       1443      2403
hdisk8          36.2     568.0      27.4       1185      1655


VSD server sp21n13
------------------
tty:      tin         tout   avg-cpu:  % user    % sys     % idle    % iowait
          0.0        171.1               0.8     11.2       22.4      65.7

(We only show you the three disks/VSDs for which this node is
VSD primary server)

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk3          31.3     568.5      24.8       1273      1575
hdisk4          29.1     598.4      23.0       2156       842
hdisk2          43.7     934.9      31.9       2489      2195
```

- vmstat

```
vmstat
------

kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 2  1 75398 15982   0   0   0   0    0   0 923 7824 4678 25 25 50  0
 3  1 75398 15992   0   0   0   0    0   0 921 6781 4051 22 23 55  0
 1  1 75396 15984   0   0   0   0    0   0 895 5774 3275 21 19 60  0
 3  1 75396 15983   0   0   0   0    0   0 947 7728 4455 25 26 49  0
 2  1 75395 15985   0   0   0   0    0   0 897 5737 3126 21 19 60  1
 1  1 75395 15988   0   0   0   0    0   0 886 4919 2580 20 16 64  0
 2  1 75395 15984   0   0   0   0    0   0 910 6059 3425 22 20 58  0
 2  1 75395 15988   0   0   0   0    0   0 868 5113 2877 20 18 62  0
 3  1 75395 15985   0   0   0   0    0   0 900 6477 3672 23 21 56  0
 2  1 75395 15986   0   0   0   0    0   0 860 5123 2769 19 16 65  0
 2  1 75395 15998   0   0   0   0    0   0 864 5380 3117 18 19 63  0
 2  1 75395 15989   0   0   0   0    0   0 930 6558 3883 23 22 55  0


Note.
We get this output from vmstat for one minute during all the test period:
(just mention it !!)

kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
35  1 75847 15505   0   0   0   0    0   0 1046 9454 12656 29 53 18  0
37  1 75847 15498   0   0   0   0    0   0 1030 10439 11668 30 50 20  0
15  1 75845 15499   0   0   0   0    0   0 1047 9069 10004 28 44 28  0
12  1 75845 15499   0   0   0   0    0   0 1034 10148 9860 33 43 25  0
 4  1 75845 15499   0   0   0   0    0   0 1146 11037 8345 31 38 31  0
```

- sar

```
sar -P ALL
----------

21:49:15  0       19      15       0      66
          1       18      17       0      65
          2       13      12       0      75
          3       13      12       0      75
          4       13       9       0      79
          5        9       9       0      82
          -       14      12       0      74
21:49:25  0       23      24       0      53
          1       23      22       0      55
          2       24      18       0      58
          3       18      18       0      64
          4       17      14       0      69
          5       17      14       0      69
          -       20      18       0      61
```

- tprof

```
tprof -k -x sleep 300
---------------------

(Initial section file __prof.all)
   Process      PID      TID    Total   Kernel     User   Shared    Other
   =======      ===      ===    =====   ======     ====   ======    =====
      wait     1806     1807    21108    21108        0        0        0
      wait     1548     1549    19735    19735        0        0        0
      wait     1290     1291    19054    19054        0        0        0
      wait     1032     1033    18113    18113        0        0        0
      wait      516      517    17212    17212        0        0        0
      wait      774      775    17086    17086        0        0        0
    update    23840    23013     3223     1414        1     1808        0
     mmfsd    21750    18299     2354     1191     1117       46        0
     tprof    29112   248533     1707      375     1270       62        0
    router    23326    67923     1252      584        1      667        0
     mmfsd    21750   110495     1082      647      349       86        0
     mmfsd    21750   179953     1055      591      386       78        0
     mmfsd    21750    29741      972      551      354       67        0
     mmfsd    21750   109463      941      571      321       49        0


(Summary section file __prof.all)
          Process     FREQ    Total   Kernel     User   Shared    Other
          =======      ===    =====   ======     ====   ======    =====
             wait        6   112308   112308        0        0        0
           server      740    41074    17816      428    22830        0
            mmfsd       57    27136    16482     9120     1534        0
           update        1     3223     1414        1     1808        0
           router        2     1798      789        4     1005        0
            tprof        1     1707      375     1270       62        0
              gil        4     1657     1657        0        0        0
            hatsd        3      891      742       85       64        0
            trace        1      270      270        0        0        0
          swapper        1      165      165        0        0        0
             sadc        1      132      128        4        0        0
```

- tprof (cont.)

```
(Subroutine section file __prof.all)
           Subroutine   Ticks     %           Source   Address  Bytes
         =============  ======  ======         =======   =======  =====
            .waitproc  108526    56.9 ../../../../../src/bos/kernel/proc/dispatc
h.c     60576    244
              .xbcopy    5204     2.7          misc.s    149468    284
      .e_block_thread    1559     0.8 ../../../../../src/bos/kernel/proc/sleep2.
c    174976   1088
     .ep_block_thread    1363     0.7 ../../../../../src/bos/kernel/proc/sleep3.
c     55992   1064
         .exbcopy_ppc    1149     0.6      misc_ppc.s    748736    188
        .disable_lock    1124     0.6           low.s     36868    764
              .kwakeup     891     0.5 ../../../../../src/bos/kernel/proc/sleep.c
    157788    408
       .simple_unlock     857     0.4           low.s     39168    512
             .sc_flih     844     0.4           low.s     14024    260
       .threaddata_sc     769     0.4           low.s     16996   7580
         .simple_lock     751     0.4           low.s     38144   1024
              .xmalloc     555     0.3 ../../../../../src/bos/kernel/ldr/xmalloc.
c   1064104   1604
       .thread_unlock     547     0.3 ../../../../../src/bos/kernel/proc/sleep3.
```

GPFS configuration with wide node

Following are summaries of system monitoring during various tests when running 750 users (taken from peak time):

This screen contains the monitoring data of the wide node during the error period.

- vmstat

```
vmstat 2
--------

kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
110  0 67533   140   0   1  12  12   12   0 291 3224 7223 62 38  0  0
302  0 67544   130   0   3   8   8   12   0 321 4199 6961 67 33  0  0
15  0 67563   127   0   4  10  12   13   0 277 4081 5356 64 36  0  0
18  0 67615   127   0   2  26  28   41   0 310 4236 7171 64 36  0  0
24  0 67623   129   0   3   8   8    8   0 314 4217 6160 64 36  0  0
43  0 67662   124   0   7  22  24   28   0 304 4598 7102 62 38  0  0
15  0 67672   140   0   3  13  16   18   0 311 3223 7804 58 42  0  0
295  0 67676   136   0   0   0   0    0   0 314 3293 9989 58 42  0  0
299  0 67680   138   0   1   3   4    4   0 300 3562 8449 65 35  0  0
292  1 67711   131   0   0   7  12   12   0 350 3721 7097 53 47  0  0
284  0 67753   136   0   0  11  24   25   0 341 3502 9667 58 42  0  0
283  0 67780   128   0   2   6  12   13   0 318 3492 6423 64 36  0  0
272  0 67788   136   0   0   4   8   12   0 318 3375 7596 60 40  0  0
23  0 67792   131   0   0   0   0    0   0 303 2998 8410 52 48  0  0
10  0 67796   133   0   1   3   4    4   0 382 3203 6743 62 38  0  0
15  0 67811   134   0   0   8   8    8   0 353 2801 9670 56 44  0  0
19  0 67818   134   0   0   4   4    4   0 340 2578 7393 62 38  0  0
27  0 67822   134   0   2   4   4    4   0 341 2811 8122 56 44  0  0

Note:
In the run queue you can see the mmfsd threads and the Lotus Notes threads.
```

- tprof

```
tprof -k -x sleep 300
---------------------
(Initial section file __prof.all)
          Process      PID      TID    Total   Kernel     User   Shared    Other
          =======      ===      ===    =====   ======     ====   ======    =====
            mmfsd    19996    23021      449      186      244       19        0
            mmfsd    19996    12087      384      180      189       15        0
            mmfsd    19996    27691      352      158      179       15        0
            mmfsd    19996    26689      259      111      135       13        0
            mmfsd    19996   134385      249      110      130        9        0
            mmfsd    19996    20517      246       99      137       10        0
            mmfsd    19996    21575      224       89      124       11        0
            hatsd    14168    15471      213      176       31        6        0
           update    18888    38319      207       69        0      138        0
           router    10438    66657      199       89        0      110        0

(Summary section file __prof.all)
        Process     FREQ    Total   Kernel     User   Shared    Other
        =======      ===    =====   ======     ====   ======    =====
         server      512     7583     3623       44     3916        0
          mmfsd       22     3331     1538     1640      153        0
         router        2      313      120        2      191        0
          hatsd        3      233      196       31        6        0
         update        1      207       69        0      138        0
      PID.25012        1      168       27      141        0        0
            gil        4       53       53        0        0        0
        swapper        1       25       25        0        0        0
           amgr        3       23       12        0       11        0
          trace        1       20       20        0        0        0
         adminp        1       15        4        0       11        0

Note:
The processes: server, update, router, amgr, sched and adminp, are Lotus
Notes server processes.
```

- tprof (cont.)

```
(Subroutine section file __prof.all)
          Subroutine   Ticks     %           Source   Address  Bytes
        =============  ======  ======         =======   =======  =====
       .unlock_enable    2165    18.0           low.s     37636    508
              .xbcopy     563     4.7          misc.s    194724    188
             .trchook     152     1.3        trchka.s     63868    468
       .threaddata_sc     125     1.0           low.s     16844   7732
            .sc_flih     116     1.0           low.s     14004    240
           .sys_call      84     0.7           low.s     14244    392
              .lockl      82     0.7           low.s     44224    512
             .unlockl      64     0.5           low.s     45312   1712
        .disable_lock      62     0.5           low.s     36868    764
              .xmfree      51     0.4 ../../../../../src/bos/kernel/ldr/xmalloc.
c    938372   1812
          .v_relpages      48     0.4 ../../../../../src/bos/kernel/vmm/POWER/v_
relsubs.c    110268    588
             .xmalloc      40     0.3 ../../../../../src/bos/kernel/ldr/xmalloc.
c    941308   1608
         .simple_lock      38     0.3           low.s     38144   1024
      .e_block_thread      36     0.3 ../../../../../src/bos/kernel/proc/sleep2.
c    168820   1044
       .simple_unlock      33     0.3           low.s     39168    512
                  .cs      31     0.3           low.s     13408    416
```

Appendix F. How to Get the Examples in This Book

The examples in this publication are available through the ITSO FTP site or at the World Wide Web.

F.1 FTP Site

The files are available for anonymous FTP from www.redbooks.ibm.com. To retrieve the files using FTP, you must have access to the Internet. If you do, use the following procedure:

```
#
# mkdir /usr/local
# cd /usr/local
# ftp www.redbooks.ibm.com
ftp> bin
ftp> cd /redbooks/SG245165
ftp> get sg245165.tar.Z
ftp> quit
#
# uncompress sg245165.tar.Z
# tar xvf sg245165.tar
rm sg245165.tar
#
```

Figure 44. Installing Examples to Recommended Location Using FTP

F.2 WWW Site

The examples can also be downloaded using the World Wide Web. The URL www.redbooks.ibm.com provides details on the procedure.

Note:

These examples have been tested on pre-release versions of GPFS. They may not be suitable for use in a particular environment, and are not supported by IBM in any way. IBM is not responsible for your use of these examples.

F.3 LiSt Open File

```
        lsof (LiSt Open Files) version 4
            (revision 4.16)


***********************************************************************
| The latest release of lsof is always available via anonymous ftp |
| from vic.cc.purdue.edu.  Look in pub/tools/unix/lsof.             |
***********************************************************************


***********************************************************************
* IMPORTANT!  This README file explains how the lsof tar file is   *
* assembled -- it's a "wrapper" tar file.  Please read the       *
* explanation of its naming and construction, immediately       *
* following the initial list of supported dialects.           *
***********************************************************************


Lsof version 4 lists open files for running Unix processes.  It is
a descendent of ofiles, fstat, and lsof versions 1, 2, and 3.  It
has been tested on these UNIX dialects.

    AIX 4.1.·45“ and 4.2·.1“
    BSDI BSD/OS 2.1 and 3.0 for Intel-based systems
    DC/OSx 1.1 for Pyramid systems
    Digital UNIX (DEC OSF/1) 2.0, 3.2, and 4.0
    FreeBSD 2.1.·67“, 2.2 and 3.0 for Intel-based systems
    HP-UX 9.x and 10.20
    IRIX 5.3, 6.2,6.3, and 6.4
    Linux 2.0.30, 2.1.2·89“, 2.1.3·45“, and 2.1.42 for Intel-
        based systems
    NetBSD 1.2 for Intel and SPARC-based systems
    NEXTSTEP 3.1 for NEXTSTEP architectures
    OpenBSD 2.0 and 2.1 for Intel-based systems
    Reliant UNIX 5.43 for Pyramid systems
    RISC/os 4.52 for MIPS R2000-based systems
    SCO OpenServer Release 3.0 and 5.0.x for Intel-based
        systems
    SCO UnixWare 2.1.·12“ for Intel-based systems
    Sequent PTX 2.1.9, 4.2.1, and 4.·34“
    Solaris 2.·345“, 2.5.1, and 2.6-Beta_UpdateII
    SunOS 4.1.x
    Ultrix 4.2

Lsof 4 may work on older versions of these dialects, but hasn't
been tested there.  Lsof versions 2 and 3 are still available and
may provide older dialect version support.  See the notes on them
in this file.
```

```
The pub/tools/unix/lsof/contrib directory on vic.cc.purdue.edu also
contains information on other ports.

Version 4 of lsof is distributed as gzip'd and compressed tar archives
in the files:

    ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/lsof.tar.gz
  and
    ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/lsof.tar.Z

These files are links to the current distribution, whose name
includes the revision number:

    ftp://vic.cc.purdue.edu/pub/tools/unix/lsof_<rev>_W.tar.gz
  and
    ftp://vic.cc.purdue.edu/pub/tools/unix/lsof_<rev>_W.tar.Z

<rev> is the revision number -- e.g., 4.16.  The _W'' marks the
tar file as a wrapper -- the source tar file is inside the wrapper.
(A tar file with a .gz'' suffix is gzip'd; .Z'', compressed.)
The wrapper method is used to join instruction and PGP certificate
files with their source tar file.  The PGP certificate file
authenticates the source tar file.

When the wrapper tar is gunzip'd or uncompressed, and its tar
contents are extracted, these files are produced in the current
directory where the extraction was performed:

    00.README.FIRST         contains introductory distribution
                information.

    README.lsof_<rev>       contains instructions for the
                security-conscious on how to be
                sure that no one has tampered with
                the tar file.

    RELEASE_SUMMARY_<rev>    is this file.

    lsof_<rev>.tar        is a tar file, containing the lsof
                sources.  When extracted with tar
                it creates a subdirectory in the
                current directory where the extraction
                was performed, named lsof_<rev>.
                The lsof source files will be found in
                lsof_<rev>.

    lsof_<rev>.tar.asc    is a PGP certificate, authenticating
```

```
                the lsof_<rev>.tar file.  See the
                README.lsof_<rev> file for more
                information on PGP authentication of
                lsof_<rev>.tar.

If you've obtained this file and an lsof distribution from a mirror
site, please be aware that THE LATEST VERSION OF LSOF IS AVAILABLE
VIA ANONYMOUS FTP FROM VIC.CC.PURDUE.EDU (128.210.15.16) IN THE
PUB/TOOLS/UNIX/LSOF DIRECTORY.

(If you need a copy of gunzip, look for it at prep.ai.mit.edu in
pub/gnu.)


*  The July 22, 1997 revision (4.14): improves Solaris VxFS support;
   cleans up typos and confusion in Configure -help output; adds
   OODIALECTS, containing UNIX dialect version numbers, for use by
   Configure and the man page.

*  The August 15, 1997 revision (4.15): better aligns Configure
   -help output; improves Solaris VxFS handling; adds TCP/TPI state
   information to socket file output; removes special commands for
   Solaris 2.6 Beta test from Configure.

*  The September 25, 1997 revision (4.16): adds the reporting of
   TCP/TPI queue lengths (all dialects) and window sizes (Solaris
   only); adds the -T option to deselect or select TCP/TPI info
   reporting; fixes anomalies in socket file SIZE/OFF reporting
   for some dialects; fixes a bug in service name range handling;
   forces use of non-BSD Pyramid C compiler; adds support for Linux
   glibc2; adds information to OOFAQ; distributes Kapil Chowksey's
   <kchowksey@hss.hns.com> Perl 5 script, using lsof, that implements
   an identd server; changes IRIX 6.4 xfs_inode guess.


Read the OO.README.FIRST in the lsof distribution first.  Read the
OODIST distribution file for more details on feature additions and
bug fixes.  The OOREADME distribution file gives background and
installation information.  The OOFAQ file contains a list of
frequently asked questions and their answers.  The OODCACHE file
explains device cache file path formation.  The OOPORTING file
contains information on porting lsof to other Unix dialects.  The
OOQUICKSTART file gives q quick introduction to using lsof.  The
distribution files lsof.8 (nroff source) and lsof.man (nroff
formatted output) contain the manual page for lsof; it is the only
other documentation besides the source code (it's included).
```

Version 4 Binaries
==================

Version 4 binaries for some revisions, dialects, and platforms may
be found in pub/tools/unix/lsof/binaries. Check the README files
for exact descriptions. Check the dialect-specific Makefiles for
installation instructions. CHECKSUMS and PGP certificates are
provided for authentication.

Please think very carefully before you decide to use a binary from
this distribution instead of making your own from the sources.
Here are some points to consider:

1. Lsof must run setgid or setuid. Are you willing to trust
    that power to a binary you didn't construct yourself?

2. Lsof binaries may be generated on a system whose configuration
    header files differ from yours. Under Digital UNIX (DEC OSF/1),
    for example, lsof includes header files from the machine's
    configuration directory, /sys/<name>. Are you willing to gamble
    that your configuration directory's header files match the ones
    used to compile lsof?

3. Lsof is often configured with specific options that are
    determined from the configuration of the system on which it is
    configured -- e.g., DECnet for Ultrix, Solaris patch level,
    dynamic loader libraries, etc. Are you sure that the lsof
    binary you retrieve will have been configured for your system?
    If you get a binary that is misconfigured for you, it may not
    work at all.

If you haven't already guessed, I believe firmly that you should
retrieve sources and build your own binary. If you still want to
use the distribution binaries, please authenticate what you retrieved
with the PGP certificates; please compare checksums, too.

Version 4 Checksums
===================

Security checksums -- both MD5 and sum(1) -- for revisions of lsof
version 4 are contained in the README.lsof_<rev> files in the
wrapper tar files of pub/tools/unix/lsof.

PGP Certificates
================

```
The lsof source tar and binary files are accompanied by PGP
certificates in files that have an "asc" extension.  The certificates
are signed with my public key, which may be found in the file:

    ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/Victor_A_Abell.pgp

My key is also available via public key servers and:

    http://www-rcd.cc.purdue.edu/abe/


Old Dialect Support
===================

Remnants of source code and binaries for dialects for which lsof
once provided support may be found on vic.cc.purdue.edu in:

    ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/OLD/binaries
  and
    ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/OLD/dialects

Dialects no longer supported include:

    CDC EP/IX
    Motorola V/88
    Novell UnixWare
    Pyramid DC/OSx
    Sequent DYNIX


Vic Abell <abe@cc.purdue.edu>
September 25, 1997
```

Appendix G. Special Notices

This publication is intended to help both RS/6000 SP specialists and general users who want to make use of a parallel file system. The information in this publication is not intended as the specification of any programming interfaces that are provided by General Parallel File Systems for AIX. See the PUBLICATIONS section of the IBM Programming Announcement for General Parallel File Systems for AIX for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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 program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each

item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

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

AIX®	BookManager®
IBM®	LoadLeveler®
NetView®	POWERparallel®
PowerPC 604	PROFS®
RS/6000	SP
SP2®	System/390®

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

Appendix H. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

H.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 225.

- *RS/6000 SP PSSP 2.3 Technical Presentation*, SG24-2080
- *RS/6000 SP PSSP 2.2 Technical Presentation*, SG24-4868
- *RS/6000 SP High Availability Infrastructure*, SG24-4838
- *RS/6000 SP PSSP 2.2 Survival Guide*, SG24-4928
- *RS/6000 SP Monitoring: Keeping It Alive*, SG24-4873
- *RS/6000 SP: Problem Determination Guide*, SG24-4778
- *RS/6000 SP System Management: Easy, Lean, and Mean*, GG24-2563

H.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

H.3 Other Publications

These publications are also relevant as further information sources:

- *IBM General Parallel File System for AIX: Installation and Administration Guide*, SA22-7278 (in press)
- *IBM Parallel System Support Programs for AIX: Managing Shared Disks*, SA22-7279

- *IBM Parallel System Support Programs for AIX: Administration Guide*, GC23-3897
- *IBM Parallel System Support Programs for AIX: Installation and Migration Guide*, GC23-3898
- *IBM Parallel System Support Programs for AIX: Diagnosis and Messages Guide*, GC23-3899
- *IBM Parallel System Support Programs for AIX: Command and Technical Reference*, GC23-3900
- *IBM Parallel System Support Programs for AIX: Event Management Programming Guide and Reference*, SC23-3996
- *IBM Parallel System Support Programs for AIX: Group Services Programming Guide and Reference*, SC28-1675
- *AIX Versions 3.2 and 4 Performance Tuning Guide*, SC23-2365

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com/.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type the following command:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/redbooks/

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com/
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number
- Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

List of Abbreviations

ACL	access control list
AIX	Advanced Interactive Executive
AMG	adapter membership group
ANS	abstract notation syntax
APA	all points addressable
API	application programming interface
BIS	boot-install server
BSD	Berkeley Software Distribution
BUMP	bring-up microprocessor
CP	crown prince
CPU	central processing unit
CSS	communication subsystem
CW	control workstation
DB	database
EM	event management
EMAPI	Event Management Application Programming Interface
EMCDB	Event Management Configuration Database
EMD	Event Manager Daemon
EPROM	erasable programmable read-only memory
FIFO	first in - first out
GB	gigabytes
GL	group leader
GPFS	General Parallel File System for AIX
GS	group services
GSAPI	Group Services Application Programming Interface
GVG	global volume group
hb	heart beat
HPS	High Performance Switch
hrd	host respond daemon
HSD	hashed shared disk
IBM	International Business Machines Corporation
IP	Internet Protocol
ISB	intermediate switch board
ISC	intermediate switch chip
ITSO	International Technical Support Organization
JFS	journal file system
LAN	local area network
LCD	liquid crystal display
LED	light emitter diode
LRU	least recently used
LSC	link switch chip
LVM	logical volume manager
MB	megabytes
MIB	management information base
MPI	message passing interface
MPL	message passing library
MPP	massively parallel processors
NIM	network installation manager
NSB	node switch board
NSC	node switch chip

OID	object ID
ODM	object data manager
PE	parallel environment
PID	process ID
PROFS	Professional Office System
PSSP	Parallel System Support Program
PTC	prepare to commit
PTPE	Performance Toolbox Parallel Extensions
PTX/6000	Performance Toolbox/6000
RAM	random access memory
RCP	remote copy protocol
RM	resource monitor
RMAPI	Resource Monitor Application Programming Interface
RPQ	request for product quotation
RSI	remote statistics interface
RVSD	recoverable virtual shared disk
SBS	structured byte string
SDR	System Data Repository
SMP	symmetric multiprocessors
SNMP	system network management protocol
SPDM	SP Data Manager
SPMI	System Performance Measurement Interface
SRC	system resource controller
SSI	single system image
TS	topology services
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
VSD	virtual shared disk
VSM	visual system management

Index

Special Characters

A

B

C

D

E

F

G

H

I

J

L

M

S

T

U

V

W

ITSO Redbook Evaluation

GPFS: A Parallel File System
SG24-5165-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@us.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

SG24-5165-00
Printed in the U.S.A.