IBM

Configuration and Tuning GPFS for Digital Media Environments

- An introduction to Digital Media environment
- GPFS configuration internals revealed
- Installing and tuning your environment

Octavian Lascu
Andreas Lüngen
Dag Wieers
Dino Quintero

Redbooks

ibm.com/redbooks

International Technical Support Organization

Configuration and Tuning GPFS for Digital Media Environments

November 2005

SG24-6700-00

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

First Edition (November 2005)

This edition applies to Version 2, Release 3, of IBM General Parallel File System (GPFS), (product number 5765-F64).

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

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

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

@server®	BladeCenter®	POWER5™
eServer™	Blue Gene®	Redbooks™
pSeries®	DB2®	Redbooks (logo) ™
xSeries®	Enterprise Storage Server®	RS/6000®
AFS®	HACMP™	Tivoli®
AIX 5L™	IBM®	TotalStorage®
AIX®	POWER™	WebSphere®

The following terms are trademarks of other companies:

IPC, Java, Sun, Ultra, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

i386, Intel, Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

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

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

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

Preface

This IBM® Redbook discusses the architecture and applications to provide an understanding of the performance impacts that GPFS components have on applications in a digital media environment, and how to avoid pitfalls and to take advantage of GPFS's parallel features.

This redbook has a step-by-step guide for configuring a GPFS cluster on Linux® on pSeries®, and a tuning section that lists and explains parameters and software components sorted by their impact on performance and availability, with special considerations for digital media solutions.

The book provides guidance and advice about how to implement GPFS in a digital media environment and how to choose the best configuration for each particular implementation. For those of you looking for a deeper understanding of GPFS, you will find this book very helpful and detailed. Although this book does not provide extensive problem determination for GPFS, it contains many hints, tips, and work-arounds for common problems or misconfigurations.

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.

Octavian Lascu is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide in all areas of pSeries and Linux clusters. Before joining the ITSO, Octavian worked in IBM Global Services Romania as a Software and Hardware Services Manager. He holds a master's degree in Electronic Engineering from the Polytechnical Institute in Bucharest and is also an IBM Certified Advanced Technical Expert in AIX/PSSP/HACMP™. He has worked with IBM since 1992.

Andreas Lüngen is an IT Specialist as member of the EMEA Advanced Technical Support (ATS) Team for storage, including presales and consulting tasks. He is the Leader of the EMEA Broadcast Innovation Center (BIC) located in Mainz, Germany. He has 10 years of experience in the IT industry, starting in a small IT company, collecting various other experiences within a midsize company (system integrator), and finally joined IBM in 1999. He holds a degree in Electrical Engineering from the University of Hannover. His areas of expertise include Linux, open systems storage, GPFS, and digital media.

Dag Wieers is a Linux and Open Source IT Specialist at IBM Belgium. He worked at IBM Belgium for five years as a System Engineer in Strategic Outsourcing and as a Consultant for Integrated Technology Services. Prior to joining IBM, he worked five years as an independent Linux Consultant for various companies. Dag has experience in automating system administration tasks, analyzing system problems, and designing Open Source solutions. Dag dove into Digital Media more than a year ago, and has augmented his knowledge considerably.

Dino Quintero is a Consulting IT Specialist at the ITSO in Poughkeepsie, New York. Before joining the ITSO, he worked as a Performance Analyst for the Enterprise Systems Group and as a Disaster Recovery Architect for IBM Global Services. His areas of expertise include disaster recovery and pSeries clustering solutions. He is certified in pSeries system administration and pSeries clustering technologies, and also an IBM Senior Certified Professional in pSeries technologies. Currently, he leads technical teams delivering IBM Redbook solutions in pSeries clustering technologies and delivering technical workshops worldwide.

Special thanks to **Mira Peltomaki**, IBM Belgium, for the outstanding contribution to this project.

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

Steven French
IBM Austin

Glenn Sullivan
IBM Beaverton

Leonard Degollado
IBM Gaithersburg

James K. McDonough
IBM Portland

Luc Andries
VRT Belgium

Ulf Troppens, Veronica Megler, Sven Öhme, Jochen Bergdolt
IBM Germany

ADMIRA project team
IBM Germany

Volker Lendecke
SerNet, Germany

Francois-Xavier Drouet, Francois Postaire
IBM France

Rick Koopman
IBM Netherlands

GPFS research team:

Frank Schmuck, Dan McNabb
IBM San Jose

GPFS development and field verification team:

Gordon McPheeters, Puneet Chaudhary, Stephen Duersch, Kuei-Yu Wang-Knop, Bruce Hempel, Lyle Gayne
IBM Poughkeepsie

Bill Hartner
IBM Austin

ITSO Editing team:

Gabrielle Velez
International Technical Support Organization

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 905
  11501 Burnet Road
  Austin, Texas 78758-3493

1

Introduction to digital media

This chapter provides an introduction to the digital media concepts and processing environments. The topics covered are:

- An overview of digital media (definition, terminology, concepts)
- The differences between streaming and IT traffic
- Examples of various digital media environments with the corresponding data access profiles
- Overview of digital media complementary offerings

1.1 What is digital media?

Digital media is any type of unstructured (non text, non numeric) rich media object, such as a video or audio clips, graphics, images, or e-mail, or a combination of these.

Today in a media environment we are facing the convergence of two key trends:

- The growing demand for rich media content
- The exploding availability of affordable bandwidth.

This is causing enterprises and organizations of all types to evaluate how they can more efficiently and strategically create, manage and distribute digital media.

The digital media covers the following aspects:

  - Content creation (harvesting, pre-processing, etc.)
  - Content management (post-processing, classification, archiving, etc.)
  - Content distribution (Internet, digital radio and TV, movies, pervasive devices, etc.)

The growing demand for compelling content, affordable bandwidth, and advancing network technologies are prompting organizations to evaluate how they can efficiently manage the digital media content throughout its entire lifecycle (see Figure 1-1 on page 3). Efficient digital media management can be used to:

- Increase the media impact on audiences
- Enhance user experiences
- Improve productivity
- Tighten security
- Reduce costs
- Contents reutilization
- Generate new revenue

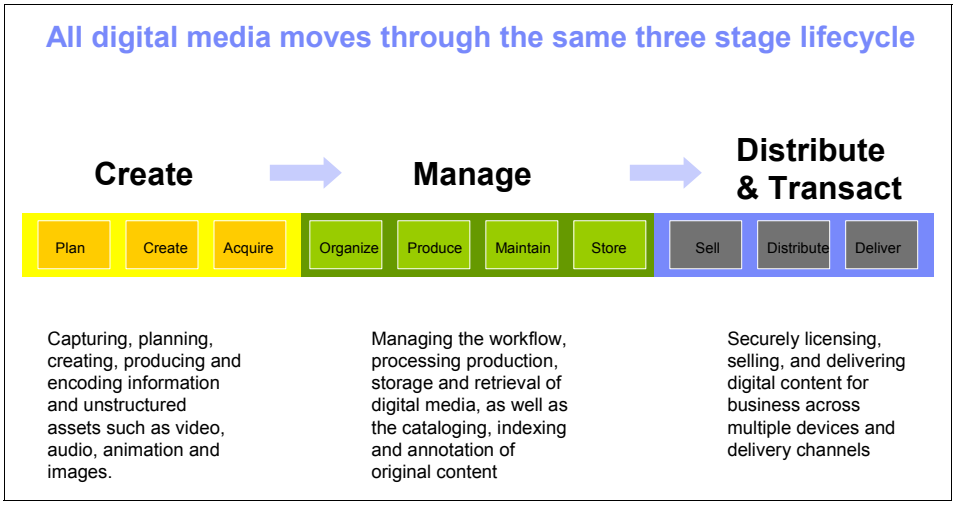

Figure 1-1 The three life cycles of digital media

As the customer requirements may be very diverse, IBM offers a broad range of Digital Media solutions for various industry segments:

- Media and entertainment
- Banking
- Government
- Publishing
- Retail
- Telecommunications

Here are the current IBM Digital Media Solutions for media and entertainment:

- Digital content creation solution for media and entertainment from IBM
- Digital content management for media and entertainment from IBM
- Digital media delivery solution for media and entertainment from IBM
- Framework for broadcasting solution from IBM
- IBM content services solution

For more information about IBM Digital Media Solutions refer to:

`http://www.ibm.com/solutions/digitalmedia`

In addition, Figure 1-2 on page 4 complements the previous information by showing the three solution segments with solution examples.

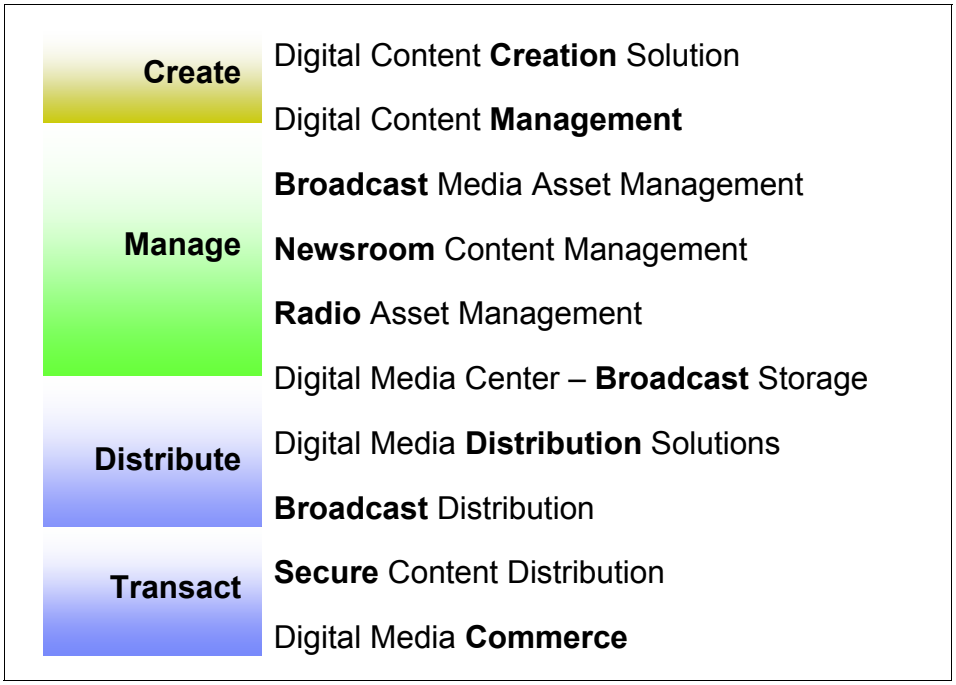

Figure 1-2 The three key solution segments with examples

All offerings are based on a customer survey and integrated into the Digital Media Framework (as a result of the survey). We describe the IBM Digital Media Framework in the following paragraph.

1.2 IBM Digital Media Framework

IBM has focused to extract the common elements from thousands of customer engagements into a solutions framework, comprising of integrated technologies from IBM and its partners. The consulting and integration skills in digital media leverage existing and evolving business solutions. This requires the understanding of the links between the various business systems that handle digital media. As the links are open, so is the entire framework. Figure 1-3 on page 5 shows the IBM Digital Media Framework.

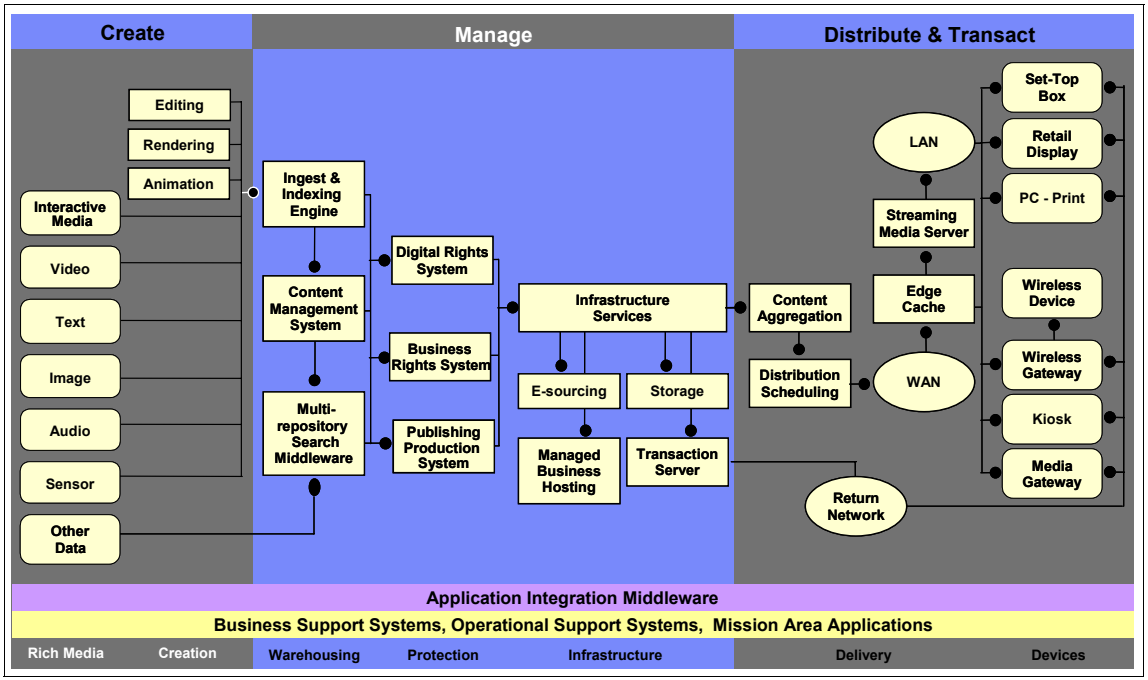

Figure 1-3 IBM's Digital Media Framework

The Digital Media Framework describes an open "ecosystem" with widely available specifications for formats, protocols, and APIs that facilitate solutions based on IBM's and business partners' components. It provides customers with the assurance of a solid foundation when dealing with heterogeneous platforms and applications (which is almost always the case in digital media). This enables customers to implement a robust set of media-based solutions tailored to solve their business problems, while taking advantage of the already installed infrastructure.

The component nature of the framework, based on open and de facto standards, eases the integration of media functions into the business.

Note: The fact that the framework is open, drives efficiency, cost savings and creativity.

An open framework is more efficient and drives better functionality for the business by removing barriers that keep rich media contents from crossing platforms. The transition from vertically integrated solutions to component-based solutions results in durable investments for customers. As needs change, individual components can be replaced without sacrificing crucial infrastructure

investments. And integration of digital media into an open IT infrastructure reduces cost.

As previously shown, there is a huge variety of environments that digital media solutions need to address. We now give some examples for access profiles in certain digital media environments.

1.3 Digital media environment examples

For a better understanding of the digital media requirements, we show some examples with their characteristics. In this section we list the data access profiles for the following application topics (which will be called from now on “profiles”, ordered by bandwidth requirements - starting with the lowest):

- Close-circuit television (CCTV)
- Video on demand (VoD)
- Broadcasting
- Post-production (comparable to broadcasting, but with much higher throughput/bandwidth requirements)

The profiles will handle the streaming rate, the client access behavior, the number of occurrences in small, medium and large environments, the read/write data ratio, and finally some special remarks for each environment.

Note: The profiles presented in the following sections are created to show the various requirements within the digital media space. You have to keep in mind that applications, devices, or even the overall behavior may be different in your environment, therefore you always need to analyze the workflow details.

1.3.1 Closed-circuit television (CCTV)

Closed-circuit television (CCTV) is a set of cameras used for video surveillance, where all components are directly linked, i.e., via cables or wireless. It is used in airports, seaports, banks, casinos, shopping centers, streets, etc., and covers the following activities:

- Airport, seaport, and border crossing protection
- Voter fraud (biometrics)
- Reservoir protection
- Nuclear facility surveillance
- Crime deterrence and detection (shopping areas, etc.)
- Property threat protection
- Home security

Table 1-1 shows the profile for CCTV environments.

Table 1-1 Close-Circuit Television (CCTV) profile

Close-Circuit Television	
Streaming rate	0.384 / 1,5 Mbit/s
Access behavior	Sequential for writing: random for reading
Block size used (application)	Variable, depends on the application
Number of concurrent occurrences	Small 1 - 10 Medium 10 - 50 Large 50 - ...
Read/write ratio	40/60%
Remarks	Mostly write in parallel; read on-demand: reverse/fast forward seeking->random access

1.3.2 Video on demand (VoD)

Video-on-Demand's goal is to enable individuals to select the video streams from a central server for viewing on an end-user device, like television, computer screen, or any other device capable of video streaming. VoD can be used for entertainment, education, and video conferences. VoD requires a network infrastructure that can handle the (large) amounts of data required for all concurrent video streams. This contains the VoD infrastructure at the service provider, as well as the network to reach the end user.

There are two different standards used (defined by the Society of Motion Pictures Television Engineers - SMPTE):

- SDTV: Standard Definition TeleVision
  SDTV provides a quality, which can be compared to a Digital Versatile Disk (DVD). The vertical resolution of SD-TV is 480 lines, and has an aspect ratio of 4:3 on screen.
- HDTV: High Definition TeleVision
  HDTV provides a higher quality compared to SDTV. The vertical resolution scales from 720 to 1080 lines and higher. The aspect ratio (width to height) is defined as 16:9 (4^2:3^2).

Table 1-2 on page 8 shows the profile for Video on demand (VoD) environments.

Table 1-2 Video on Demand (VoD) profile

Video on demand (VoD)	
streaming rate	MPEG2 MPEG4 SD-TV: 4 - 6 Mbit/s 1.5 Mbit/s HD-TV: 8 - 15 Mbit/s 6 - ... Mbit/s
access behavior	sequential
block size used (application)	128 / 256 KB
number of concurrent occurrences	small 1-100 medium 100-500 large: 500-...
read/write ratio	9:1 (90% reads for 10% reads)
remarks	mostly read in parallel; only new material is written

1.3.3 Broadcasting

In a typical broadcasting environment you can find three distinct areas working with the high quality streams (video and audio):

- ingest stations (incoming stream)
- non-linear editing (NLE) stations (handling stream)
- play out server (play the stream out)

In addition, for any high resolution stream will be a low resolution version created (for catalog and archiving purposes). You will also find common IT tasks like backup/restore and database operations.

Analyzing and NLE system only, you typically have two up to five parallel data clips referenced at a time, and most of the action is "reading", which includes to shuttle through the material. Once you decide to render the material, the read/write ratio is almost 50/50 since an NLE system does exactly read one stream as input and writes one stream as an output (with caching in between). Looking in addition at an entire production day, you may consider that a lot of time is used by seeking for appropriate material and this is read-only activity.

In a typical NLE environment, there are multiple files open at a time, and the result is one file being written when rendered. Also, most of the edit work involves to shuttle through the content, which is heavy read activity. Table 1-3 on page 9 shows the profile for broadcasting environments.

Table 1-3 Broadcasting profile

Broadcasting	
Streaming rate	25 / 50 Mbit/s
Access behavior	Sequential or random
Block size used (application)	64 / 128 / 256 KB
Number of concurrent occurrences	Small: 1-5 Medium 5-15 Large 15-...
Read/write ratio	60/40%
Remarks	Read/write in parallel audio and video Mixed/divided depending on application

1.3.4 Post-production environments

Compared to broadcasting, the post-production (mastering and distribution) environments require much higher bandwidth, as the resolution of the content is higher. Also the access behavior in post-production environments is more sequential, since this is predefined work, which involves “shuttle-in”, then work with the editing or color correction stations. Hence, the material will be opened for read, updated on the workstation and written/rendered back to the storage. Table 1-4 shows the profile for post-production environments.

Table 1-4 Post-production profile

post-production	
streaming rate	3000-5000 Mbps (400-600 MBps)
access behavior	sequential to random
block size used (application)	256 KB
number of concurrent occurrences	small 1-3 medium 3-10 large 10-...
read/write ratio	70/30%
remarks	read/write in parallel audio and video mixed/divided depending on application.

As it is not possible in this book to cover all of the digital media requirements needed to build up a reliable, scalable and performing infrastructure, we will concentrate on broadcasting.

Within IBM this is part of the Media and Entertainment sector, and the related offering from IBM is called Digital Media Center (DMC), which is described later in 1.4.1, “IBM Digital Media Center (DMC) offering” on page 13).

1.4 Broadcasting

We are facing the fact that news and media broadcasters are now rapidly making the transition from their traditional analog systems to new digital systems.

When switching to digital environment, news broadcasters are capable to react much faster response to events (news) by using a digital media asset management system, therefore being able to provide prompt/contemporary/real time content directly related to the actual happening over different distribution channels, like TV (cable), radio, or the internet. The fastest and most accurate information provided, the broadcaster gains more advertising, therefore increasing revenue.

For media oriented broadcasters, the use of a central storage platform is the platform of choice, sharing content to work in parallel with multiple non-linear editing stations, therefore increasing the efficiency compared to a sequential workflow.

Broadcasting is just one of the topics of Digital Media and Entertainment with a specific set of requirements from which we present subset of the most common ones:

- Ingest (incoming streams)
- Content management (rights- and metadata management)
- Non-Linear Editing (NLE).
- Pre-cut stations (low resolution browsing) to preview and select material
- Archiving (backup, restore, partial retrieve, space management)

Under the current cost pressure, broadcasters need also to find ways to work more efficient. In the past, typical workflows were defined as a sequential process of steps to get the final result. The following example roughly describes such an workflow (see also Figure 1-4 on page 11):

1. Capturing the material, i.e., with a movie camera
2. Bringing the material in (into a studio).
3. Copy the material (essence[1]) and create the contents metadata[2]. Here the term “metadata” is used to describe the content in more detail. This is

[1] Essence - a video file, information on a videotape or a graphics image

necessary for material contents classification (for storing in an archive), and for fast finding the desired sequence(s), without the need for browsing the entire material.

4. Storing the original material in a centralized archive (located in a room with special environment - temperature, capacity, etc.).
5. Send a copy to the cutting process, where a draft content is created
6. Redactor needs to check results (draft content - step 5). If content is ok, proceed with step 7, otherwise go back to step 5.
7. Storing the “metadata” of the new content and send it to play-out (“air” the material).
8. After transmission the material in stored in the archive (as you can see, both incoming material and the “aired” one are stored into the archive).

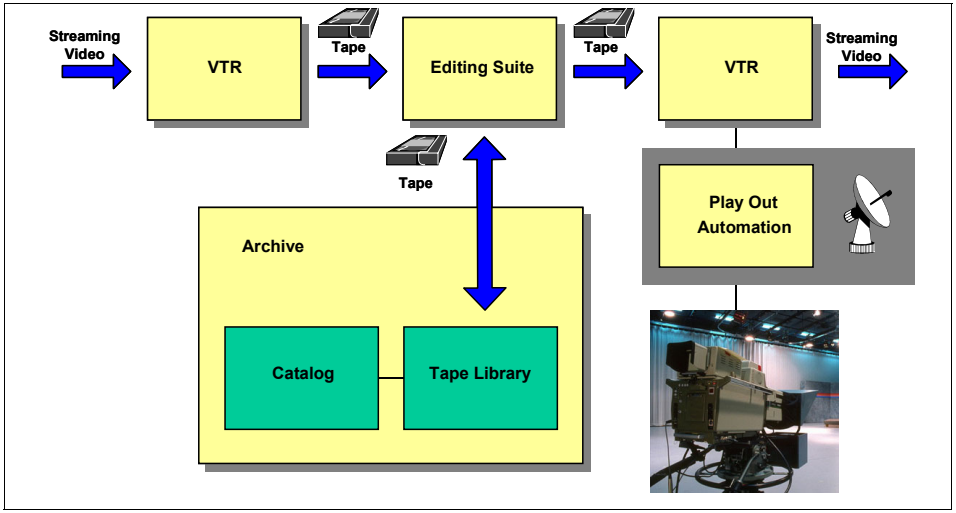

Figure 1-4 Traditional broadcaster workflow (simplified)

As you may have already observed, compared with normal IT operations, this workflow could be done much more efficiently if the process can be optimized by parallelizing the steps.

To make this happen, centralized storage is needed so that everyone has access to the same material at any given time (see Figure 1-5 on page 12). This is possible with the actual technologies, allowing broadcasters to improve the workflow in several ways:

1. Broadcasters save time (and therefore money) as the time for the overall process can be reduced dramatically by parallelizing tasks.

[2] Metadata refers to a wealth of information associated with the essence

2. By using standard IT based components instead of proprietary devices designed for a special market only (therefore expensive), broadcasters also save money.
3. This gives more flexibility to the broadcasters as the return of investment (ROI) is achieved in much shorter time, therefore reducing the overall total cost of ownership (TCO).
4. Traditional broadcasters have two different departments:
   - The IT department, familiar with IT infrastructure (provides services related to the normal business work),
   - The production department, which deals with all topics related to broadcasting.

   By merging these two departments, the resources can be freed, and can be used for more diverse tasks. This includes people as well as equipment (for example, only one tape library per location is needed).
5. In addition to the benefits related to merging the two departments, by using IBM's on demand offerings, customers have the possibility to plan and react with more flexibility to change requests. This means they can act much faster on changing customer requirements, or the demand of the marketplace.

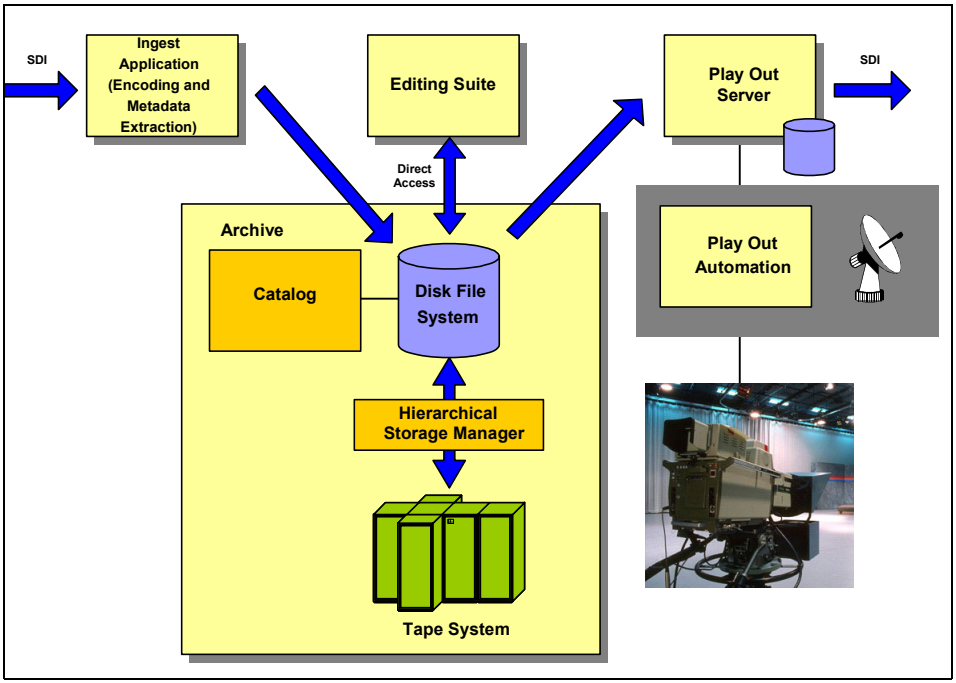

Figure 1-5 Centralized storage based broadcasting workflow (simplified)

The conclusion we can draw is that a centralized storage is the key to make the workflow more efficient and to save costs. The storage in such an environment must meet the following demands:

- Storage must be scalable
  As islands are difficult to manage and/or slowly accessible (may also need their own content management), islands must be avoided.
- Storage must perform
  It must provide the speed and bandwidth for multiple read and write streams that should be handled without interruption (concurrent access).
- Storage must be highly available and should provide for data protection
  Broadcasters need to be online at all times to reach their customers.

Since the storage is not the only component involved, we need to analyze all infrastructure components. Every component can have a big influence (impact) in the infrastructure's performance, scalability, or high availability, therefore we have to look at the following components as part of the overall design:

- Storage subsystem (controller plus attached disks)
- Storage area network (SAN)
- File system (i.e., nodes, operating system and the file system design)
- Local area network (LAN),
- Attached clients (operating system, access method, specific requirements, etc.)

The previous list contains the components of the IBM infrastructure offering, also called Digital Media Center (DMC). The following section will give a short introduction to IBM's DMC offering. To understand the components and the structure of the DMC, we also need to understand the difference between IT data and media streaming traffic. This will be explained in more detail in 1.5, "Streaming data versus normal IT traffic" on page 15.

1.4.1 IBM Digital Media Center (DMC) offering

The DMC offering may consist of the following components (the list is not comprehensive, it is just a high level overview):

- Infrastructure components; see Figure 1-6 on page 14
- Consulting services (identifying the workflow and designing the solution architecture)
- Implementation services

Note: We are using the term *offering*, as *solution* may not necessary include the architecture and design part.

This offering is based on customized components, making it very flexible to customer demands. The solution (as a result of direct customer requirements, or as resulted from the consulting activity) can be easily adapted to any special requirements, when using various applications and hardware devices.

The infrastructure has the same structure and type of components regardless if this is an entry, a medium or an enterprise size solution. It just depends on the modules chosen to fulfill the requirements for: performance, scalability, availability, and, last but not least, price.

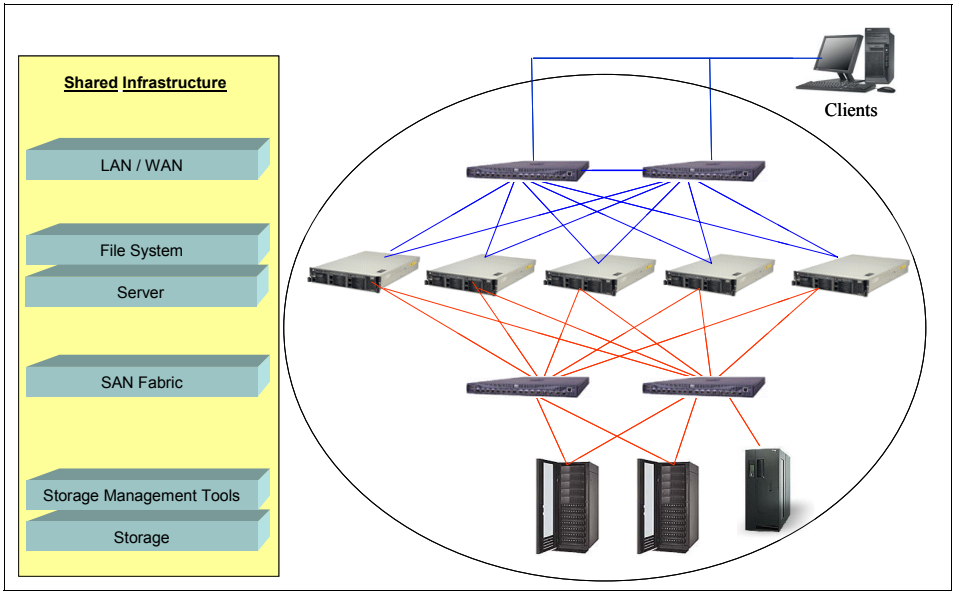

Figure 1-6 IBM offering: Digital Media Center (DMC) components

In addition to the basic components, the offering needs to be tested and tuned with/for specific applications, or for integration with third-party hardware. Often just testing the attachment capabilities of existing platforms and devices to this infrastructure may be not enough. The environment must be tested as an ensemble, under real working conditions, to make sure that the expectations are met. The most difficult part is to integrate a new devices/applications, which means to test a precise workflow, and by using all components involved.

Reminder: It is easy to attach a device to an infrastructure, but may be difficult to integrate. Most of the issues arise while integrating different products from different vendors into one solution.

The tests are usually performed in IBM labs (i.e., focussing on broadcasting), and Figure 1-7 is showing examples for tested client components of DMC.

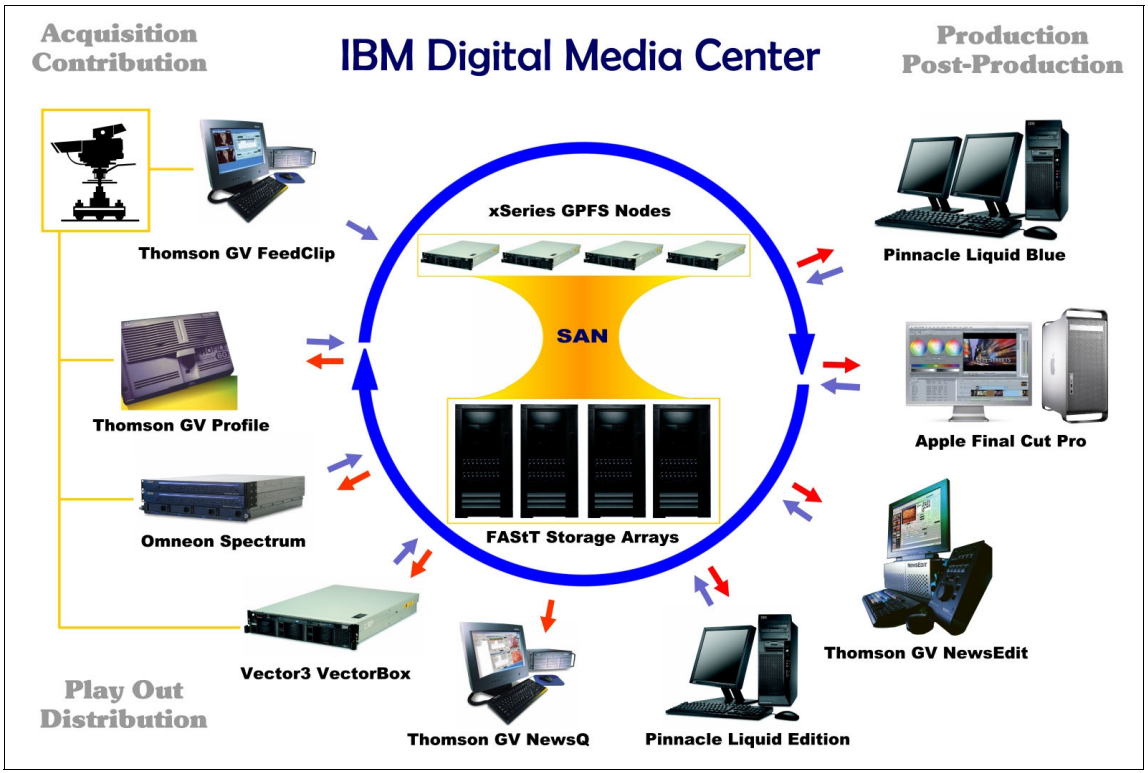

Figure 1-7 Digital Media Center -examples for tested client attachments

1.5 Streaming data versus normal IT traffic

While working within Digital Media environments we have to differentiate between two different types of data handling regarding the response time:

- **Best effort** - the common used method in the information technology (IT). Depending on the actual load of the target device, the response time may vary. Good examples are database response times, file copy, etc.

- **Streaming** - Found in Digital Media (DM) environments. More specific, this access pattern is called isochronous data streaming. According to the definition, the response time for all data packages must be the same at any point in time.

Note: “Best effort” means to request/acknowledge data packages if capacity is available to do so. Response time is variable. Normal “IT” like implementation.

“Isochronous data streaming” means to deliver a defined number of data packages within a time period. Response time is constant.

A well established (mis-)understanding is the fact that the term “Digital Media” is bounded to the term “sequential access”. For example, if you look into marketing or sales charts/papers, you will most probably find something like:

“A storage subsystem is delivering excellent performance in Digital Media environments because sequential access tests show excellent results.”

Unfortunately, this may not always be true in all Digital Media environments. To clarify this, we have to consider the following components:

- Application, which initiates the sequential stream.
- File system, which acts as an interface between the application and the storage.
- Storage subsystem, based on a collection of disks organized by a controller.

In effect, this approach is ONLY valid as long as we consider a single data stream. In most of the cases though, multiple streams are used simultaneously, making this assumption false.

Attention: As most of the storage subsystems are tuned for "overall throughput", you should be aware of the fact that not all storage devices/subsystems may be tunable/suitable/usable for data streaming.

1.6 Complementary offerings

The basic infrastructure consists of “off the shelf” components: servers, storage, network, operating system, file system and backup software. As a base, this infrastructure needs to be extended and enhanced to fulfill the requirements, therefore, additional components are needed. In the following sections we describe examples of such components, which complement the overall solution.

1.6.1 Advanced tape management with Enterprise Removable Media Manager

The Enterprise Removable Media Manager (eRMM) is a service offering for tape management which provides features deriving from similar mainframe implementations for the open systems environment. It complements the infrastructure to provide storage virtualization and advanced storage management for removable media.

In addition, eRMM can be configured as an advanced library manager for tape libraries with SCSI media changer (e.g., IBM TotalStorage® Ultra™ Scalable Library 3584) to enable tape drive sharing across heterogeneous applications.

Managing tape storage in a complex environment

Today's heterogeneous open system environments face the problem that there is no central management for removable media resources (tape). As a result, each application dealing with such removable media (e.g., backup systems) must directly interface with the physical resource (drive, library).

Therefore each application must include its own removable media management system. Hence, access control, administration and reporting of these resources is spread across multiple applications. This makes its difficult (or nearly impossible) to efficiently share libraries, cartridges and drives across heterogeneous application and library boundaries.

On the other hand, customers demand improved manageability in order to handle the rapidly growing removable storage resources. Customers want to add storage or upgrade storage without making changes to each application, also demand centralized removable media resource allocation to increase the utilization of drives, and recycling the cartridges. Centralized administration is required, as well as reporting and auditing to improve the productivity of storage management personnel.

The list of issues to be addressed in complex storage environment includes:

- No central access control, administration, and reporting.
- Media management has to be done by every application.
- Difficult or nearly impossible to share libraries, drives, and scratch pools. This is especially true in heterogeneous environments.
- Multitude of media changer and/or management interfaces (SCSI, IBM 3494, STK ACSLS/LS, ADIC, etc.)
- Adding storage or upgrading storage technologies requires changes to every application.

These issues lead to the following customer requirements:

- Improved manageability to cope with rapidly growing storage.
- Reduced total cost of ownership (TCO).
- Efficient usage of removable media storage capacity.
- Increased utilization of physical drives.
- Integrated support for vaulting[3].

To solve these issues mentioned above, eRMM provides the following features:

- Dynamic library and drive sharing across heterogeneous application boundaries and heterogeneous operating systems
- Dynamic drive and cartridge pooling
- Mount request queuing
- Policy-based drive and cartridge allocation
- Centralized access control, administration and reporting

In addition, the following features are planned for future releases:

- Policy-based media life cycle management
- Integrated off-site media management and tracking (vaulting)
- Advanced reporting and auditing

For an overview of customer requirements and faced issues and how eRMM can help, see Table 1-5.

Table 1-5 eRMM offers solutions to customer requirements and faced issues

Top customer problems/requirements	eRMM Solution
Inefficient capacity usage because tape storage is statically partitioned by application	Provide access to and sharing of a common scratch pool and dynamically resizable cartridge groups with access control
Inefficient drive utilization because its difficult to share drives (physical mapping App->Lib. Partition)	Provide dynamic sharing of tape drives including mount request queuing + additional utilization enhancements through dynamic grouping and pooling of drives
Media management has to be done by every application	Provide centralized mainframe-class media management accessible through open standards based API (IEEE1244)

[3] Vaulting: Transporting removable storage cartridges (in a coordinated manner) to/from a remote location for disaster recovery and archiving purposes.

Top customer problems/requirements	eRMM Solution
No centralized access control, administration and reporting	Provide centralized access control, administration and reporting accessible through open standards based API (IEEE1244) and SSG Console WUI
No integrated support for vaulting	Provide integrated support for manually operated libraries and vaults together with policy-based cartridge life cycle management (planned for future release)
Adding storage or upgrading storage technology requires changes to every application	Eliminate changes when storage is added and minimize impact of technology upgrades

IBM's eRMM provides a middleware between heterogeneous applications and library hardware to build a very flexible and scalable infrastructure (see Figure 1-8). It provides centralized management functions like cross library cartridge pooling, cross library drive pooling, cross application drive sharing, mount request queuing, centralized access control and auditing, and advanced reporting functions.

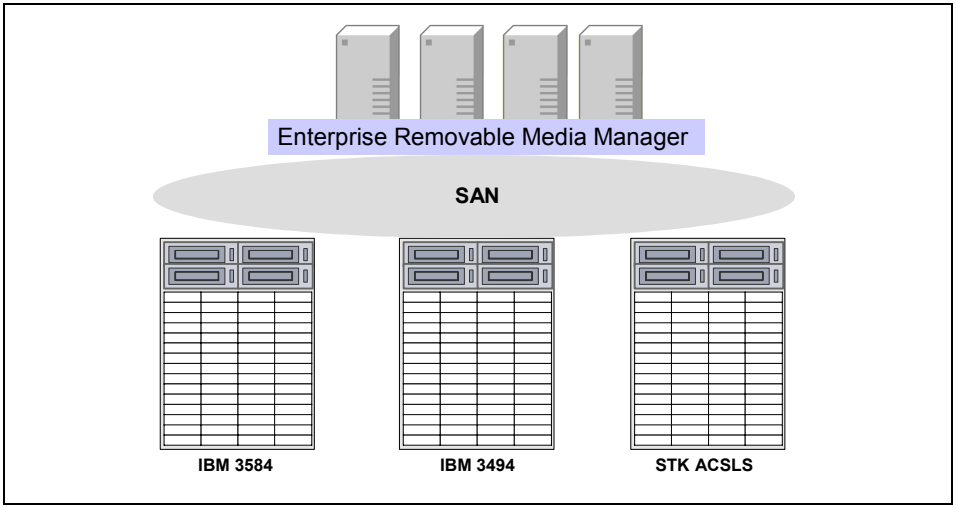

Figure 1-8 eRMM - a tape management virtualization layer

To achieve this functionality, eRMM consists of the following major components:

- Media Manager (MM)
  This is the central "server" component which among other tasks coordinates access to drives and cartridges, handles volume allocation and deallocation requests, and stores a log of all activities persistently for auditability. MM utilizes a dedicated instance of IBM DB2® for persistent storage.

- Library Manager (LM)
  The LM provides MM access to different library hardware (e.g., media changer, inventory, etc.) via an standardized interface. In this way new library hardware can be added to an already running eRMM system by just adding the respective library manager.
- Host Drive Manager (HDM):
  The HDM handles MM mount and unmount commands, among others. When the HDM starts on an application server, it reports all available drive handles (e.g., /dev/rmt3, \.\Tape2) to the MM, therefore the MM dynamically learns which drive is available on which server even when the drive configuration changes after a reboot. In addition to that, after each unmount the HDM collects statistical data of the last drive access (log sense pages) and forward them to the MM.
- ADMIN console (AC)
  The admin console is the central point for eRMM configuration and administration tasks. It offers two choices:
  - Command Line Interface (CLI).
  - Web Graphical User Interface (WebGUI).
- TSM External Library Manager (ELM) for eRMM
  The ELM enables TSM to utilize eRMM for media management purposes.

All eRMM components can run on the same or on different servers, therefore different options are available for scaling and high availability.

Application servers which need access to eRMM managed resources (libraries, drives, cartridges) must run the HDM. In addition to that TSM servers needs the ELM for eRMM.

The eRMM architecture is shown in Figure 1-9 on page 21.

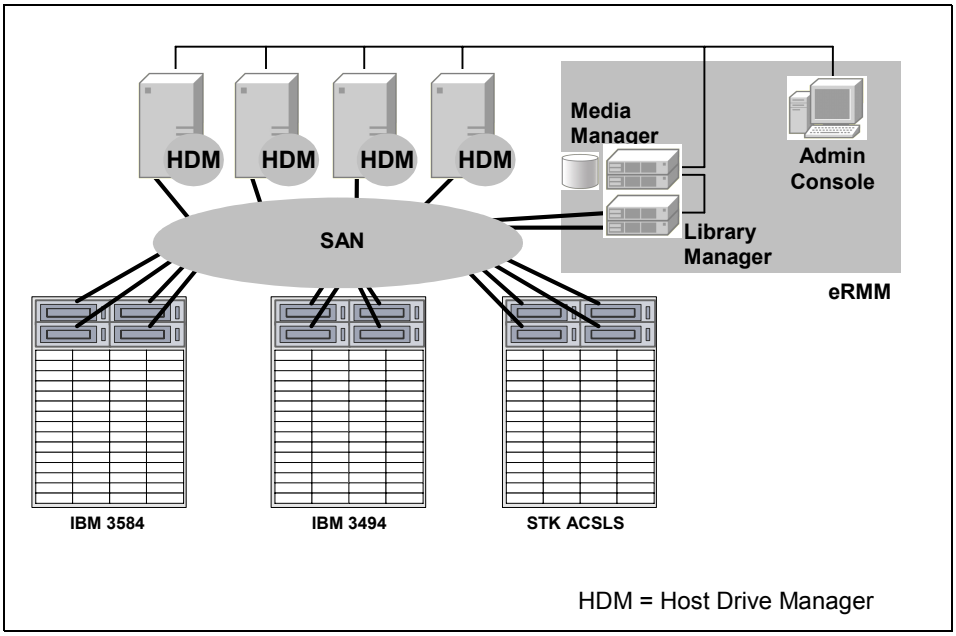

Figure 1-9 Enterprise Removable Media Manager (eRMM)

For more information about the previous topic see:

http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10358
http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10365

1.7 ADMIRA

ADMIRA is a service offering which provides media essence[4] management. It can complement the infrastructure to close the gap between normal IT based tools (like TSM) and digital media tools (like a content management application).

ADMIRA is a middleware layer that helps overcome typical problems of managing large video and audio files on IT storage infrastructures. Since large video and audio files are sometimes accompanied by descriptive metadata, low resolution versions and key frames, ADMIRA checks the consistency between all these files and performs necessary video and audio format transcodings and analysis. ADMIRA performs all media processing tasks as transactions so that rollbacks and cleanups will be performed if an error occurs. Depending on the frequency a file needs to be used, different storage platforms (online, near online, near line, and offline) and sometimes all together are all used.

[4] Essence - a video file, information on a videotape or a graphics image.

ADMIRA does not provide the complete content management system, but represents the essence manager part of it. ADMIRA is designed to receive instructions from a variety of content management or planning applications. Therefore it navigates between these applications and places files where they are needed. A typical "job" (equals a transaction) which is sent by a content management system and executed by ADMIRA is the following:

- Copy high resolution video file from ingest server to content management system
- Extract technical metadata from high resolution video file (i.e., MXF metadata)
- Archive high resolution video on tape library
- Create low resolution copy of this file
- Put low resolution version to streaming server
- Backup low resolution version to tape library
- Notify a content management system of storage locations of high and low resolution files and provide technical metadata

The execution of every task in this job can be monitored in ADMIRA. In case of an error, the complete job is seen as a transaction and the system is cleaned up during the rollback so that a failed job can be restarted.

ADMIRA is a Java™-based program which can run on Linux and AIX® and is implemented as a classic client-server model. The clients within ADMIRA environment are called agents. Examples for agents are:

- Media format conversion agents
- Media manipulation agents
- Tape quality management agent for IBM tape robotic systems

The different components of ADMIRA are described Figure 1-10 on page 23.

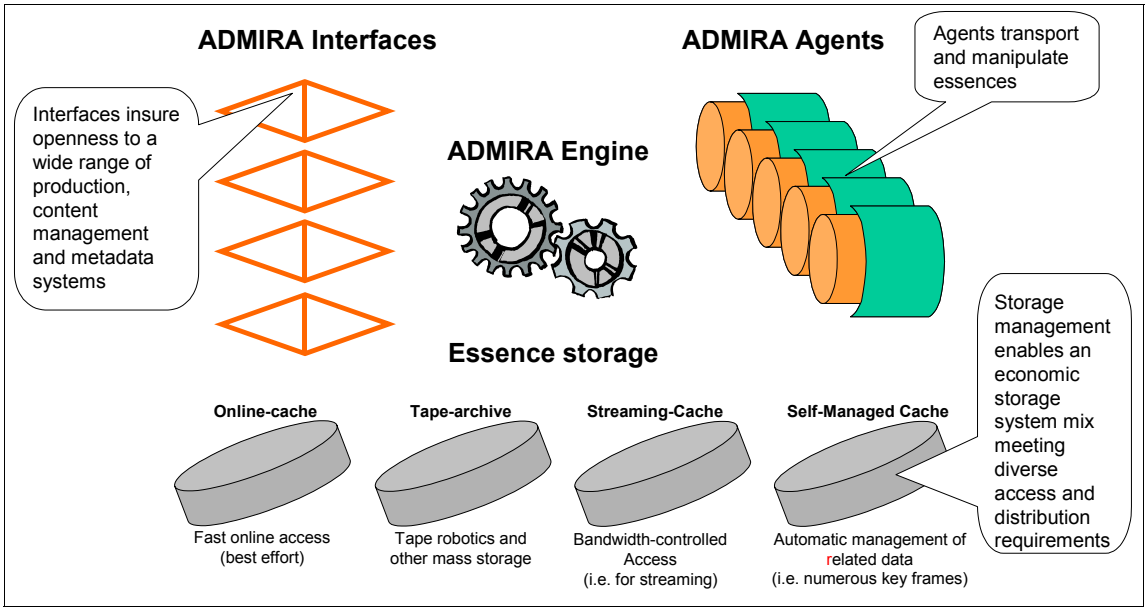

Figure 1-10 The components of ADMIRA

As ADMIRA relies on already available functions, it only requires the following additional components:

- A database - currently IBM DB2 or Oracle are supported
- A backup and restore solution - IBM Tivoli® Storage Manager (TSM)
- An application server - IBM WebSphere® Application Server (optional)

ADMIRA relies on TSM, therefore being able to address media issues, like partial file retrieve in relation to time code. The database is used to store metadata, which means in this case media essence related information, i.e., file size or file format. The database also stores the tape usage numbers (how many times a tape was mounted) retrieved from TSM, and uses this information for tape quality management functions (e.g., to determine when a physical tape needs to be replaced for avoiding losing data due to tape media failure). The Web based clients require IBM WebSphere Application Server in addition.

Figure 1-11 on page 24 shows the overall collaboration of all components involved.

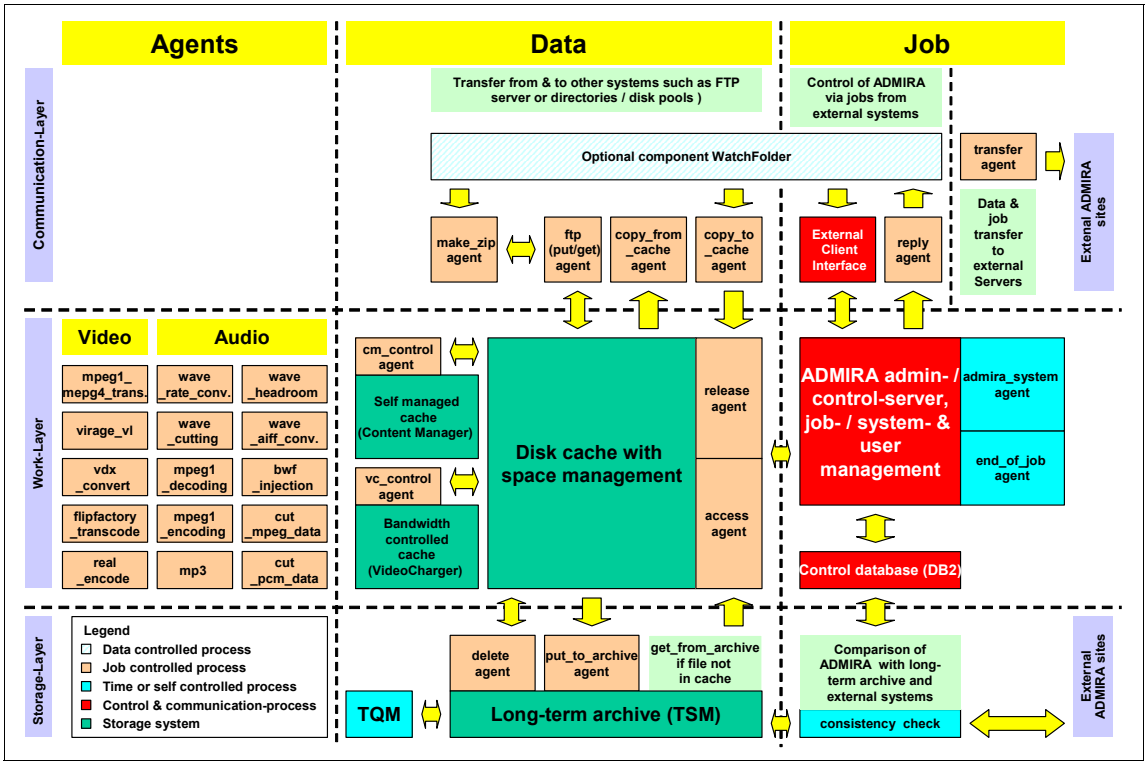

Figure 1-11 ADMIRA - architecture and workflow

Based on this client-server design, ADMIRA can also be used in distributed environments as the communication of all components is processed over the LAN, MAN or WAN (see Figure 1-12 on page 25).

Based on this client-server design, ADMIRA is a distributed system. Agents can run on different machines and platforms which are connected via LAN, MAN and WAN. New agents and new servers can be added during operation without stopping the system. This significantly increases the scalability and reliability.

In addition, several ADMIRA systems can be linked together (i.e., for broadcasters with multiple, geographically distributed sites). A *transfer agent* synchronizes the ADMIRA instances and the *consistency agent* verifies the consistency of the different instances, i.e., after a loss of network connection.

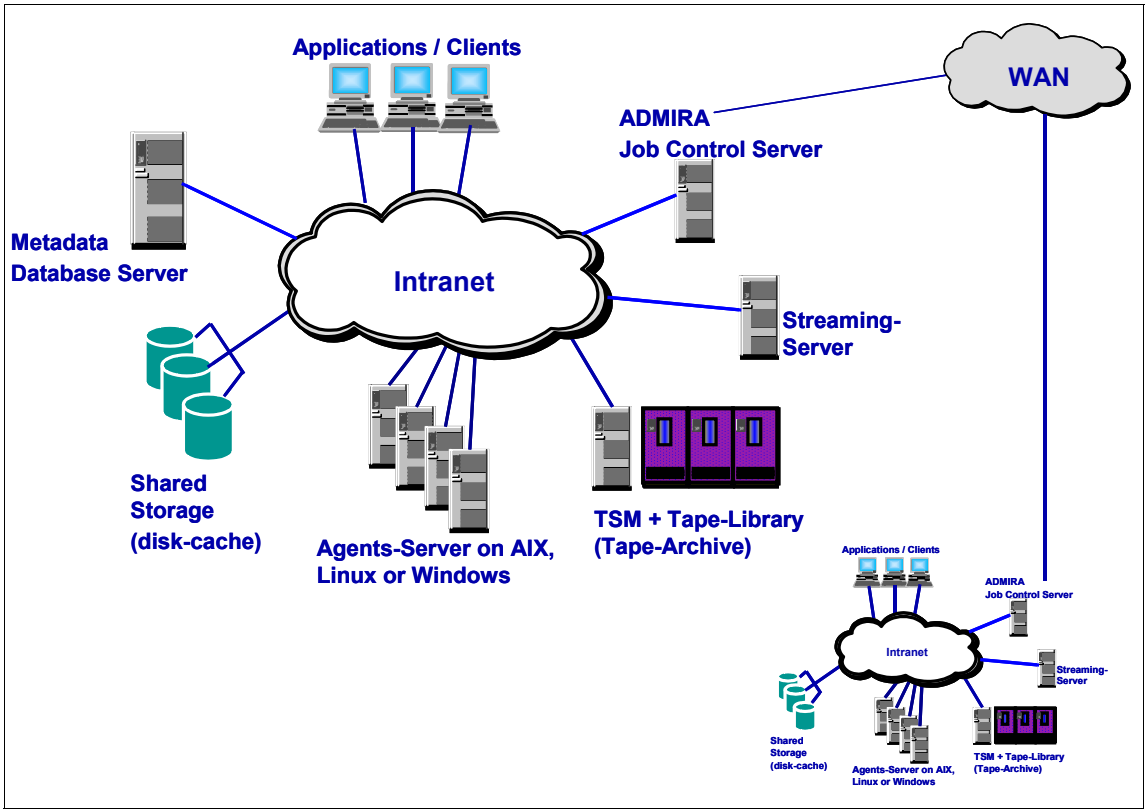

Figure 1-12 ADMIRA in distributed environments

For further information, send an email to:

`mailto:ADMIRA@de.ibm.com`

1.7.1 Open Enterprise System Virtualization (OESV)

IBM OESV is an OpenSource based, integrated solution which combines common infrastructure services as a file service (Samba) and print service (CUPS). Figure 1-13 on page 26 gives an introduction to OESV.

OESV combines external storage and server virtualization. External storage in combination with Linux-based grid functionality is mostly fault tolerant without using extra clustering software. Windows®, UNIX® and Linux file servers are bundled together into a redundant server-farm, which is exploited and reachable over virtual IP addresses, therefore providing the following advantages:

- Higher productivity and availability
- Less administration overhead

- Maintenance without downtime

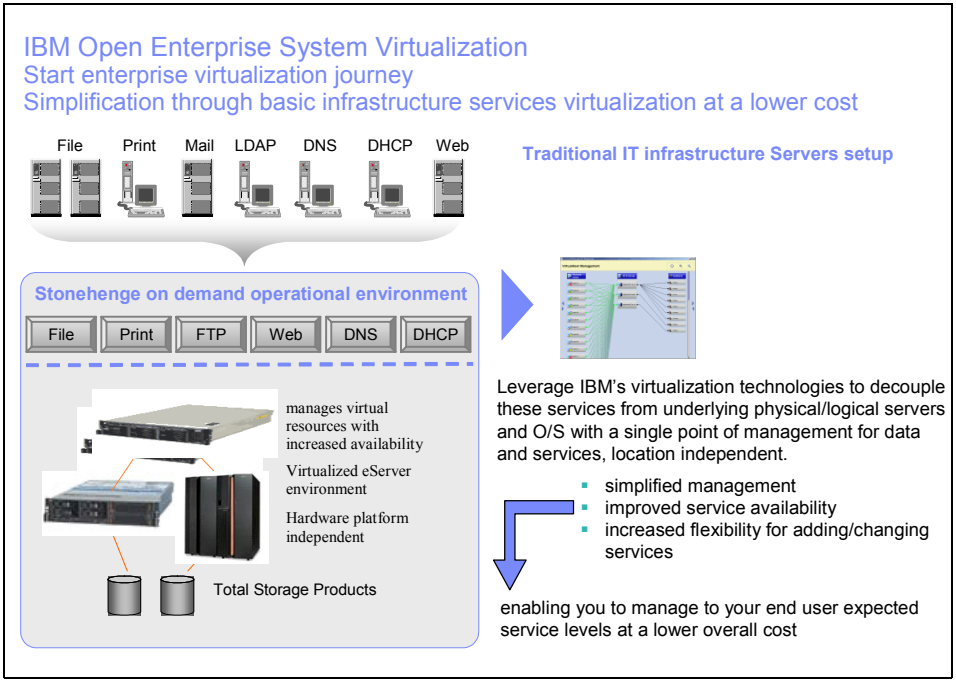

Figure 1-13 Open Enterprise System Virtualization (OESV) overview

Why use Open Enterprise System Virtualization?

In traditional environments various servers (different platforms also) hosting services are required by the end users. While the environment is growing, various shortcomings can be observed, like dependencies between servers hosting dedicated services, or available resources which are not fully used, or scaling could not be achieved without downtime. As cost pressure forces companies to use their resources more effectively, in addition to reducing costs for system management, the following requirements need to be fulfilled:

- Services should be less hardware-dependent
- Consolidate hardware and services
- Migrate and scale without disruption
- High availability and security

Figure 1-14 on page 27 shows the old approach (traditional IT services) and the goal of OESV.

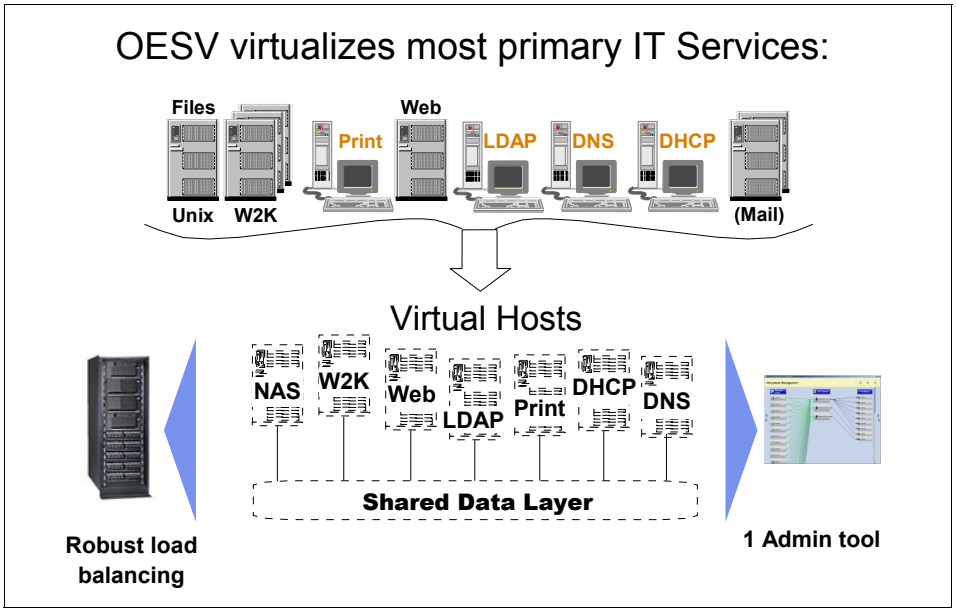

Figure 1-14 Traditional IT services versus the OESV approach

The OESV Environment splits into three layers:

- **Frontend layer**
  These are (two or more) servers, where the user connects to and gets his requests answered. The virtual host runs the virtualized services. These processes get started on the frontend (physical host). A physical host can run one or more instances of a virtual host.
- **Backend layer**
  This layer provides the "data management" and is based on OpenAFS. OpenAFS is a real distributed filesystem with features like online data replication, online migration, and transparent data access. "Transparent" in this case means that the OpenAFS client does connect to any OpenAFS server and (after successful authentication), gets the data from the file server (where the data is actually stored), but this process is handled within the OpenAFS protocol, so the application doesn't need to know where to look for the data.
- **Administration server**
  - Central instance where administrators log on with the OESV Graphical User Interface (GUI).
  - Monitors all OESV machines, both frontend and backend machines.
  - Holding miscellaneous instances:

- **LDAP**
  The OpenLDAP service installed on the Admin Server, where the accounts for the administrators are kept.
- **Database**
  Represented by the MySQL database, where among other things the logging and configuration data for the OESV environment is saved.
- **OpenAFS**
  The filesystem OpenAFS, where the actual user data is stored.

The graphical administration interface (GUI) is based on Java and is used to manage the OESV environment. This GUI can be installed on the administrators PC or a notebook, where Windows 2000 or XP are supported client operating systems. The GUI is separated in several "modules" which are used to manage the system, like Samba ("Share Module"), Logging, CUPS (Printing), Virtual Host Management, Performance Analysis and AFS® Administration.

The architecture of OESV is shown in Figure 1-15.

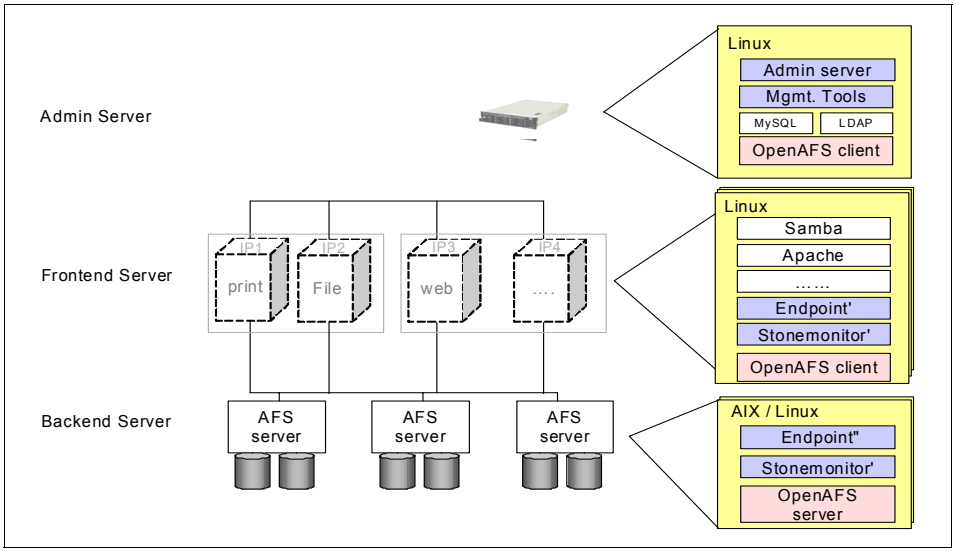

Figure 1-15 Open Enterprise System Virtualization (OESV) architecture

More info related to OESV can be found at:

http://www.ibm.com/services/de/index.wss/offering/its/a1007687

Substituting the backend server layer with the General Parallel File System (GPFS), allows for combining the advantages of both environments. OESV offers the virtualization of services, like Samba, NFS, etc., while GPFS allows the environment to scale horizontally among servers, storage and network

components. This fulfils the requirements for increased productivity, availability, less maintenance, simplified management, and also high scalability and performance demands in a digital media environment.

Figure 1-16 depicts a simplified OESV view from customer's perspective:

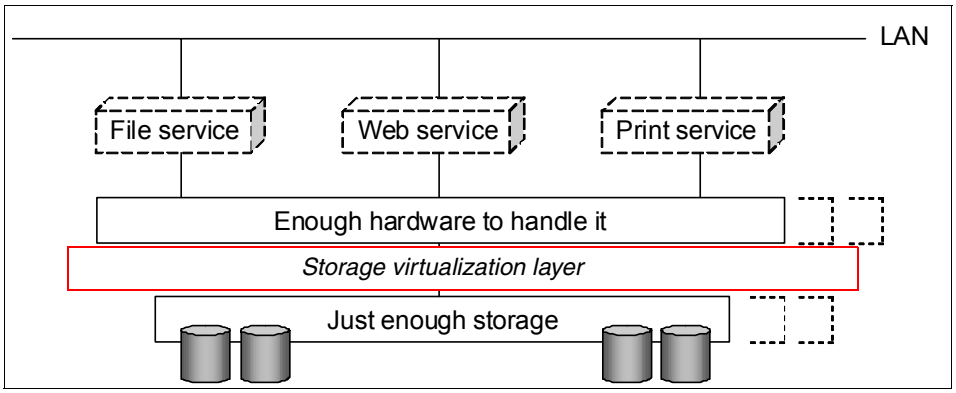

Figure 1-16 Customer requirements

2

Introduction to GPFS

This chapter provides basic as well as detailed information about the GPFS architecture, design considerations, and basic GPFS usage. This chapter covers the following topics:

- A short GPFS history
- GPFS architecture and design
- A design overview of a GPFS cluster
- A collection of tables classifying the GPFS commands

2.1 GPFS history and evolution

The early history for the IBM cluster file system (CFS) includes:

- Bell Atlantic Video on Demand Field Trial
  Shark (a predecessor of Tiger Shark) file system on an RS/6000® delivering 50 simultaneous 1.5Mbit/s MPEG streams via ADSL to 400 field trial participants in Alexandria, VA (1993-1994).
- Hong Kong Telecom Video on Demand Trial
  Tiger Shark on three RS/6000s delivering 150 simultaneous 1.5Mbit/s streams via ADSL to trial customers (1995).
- Argonne National Labs
  A Tiger Shark cluster on a 28-node SP-2 delivering 4.5Mbit/s M-JPEG video to workstation clients via an ATM WAN (MBone) using a modified version of the VIC conferencing system. This system was demonstrated at Supercomputing '95.
- Indiana University School of Music Library
  Variations Project - a digital library project using Tiger Shark on an RS/6000 to deliver digital audio and video to 130+ PC and Mac clients over an ATM network via NFS protocol.

For more details see:

`http://www.almaden.ibm.com/cs/shark/`

Originally, the Tiger Shark file system was targeted at dedicated video servers (*m*ulti*m*edia content – therefore the "mm" in all GPFS commands), initially running on standalone RS/6000 (AIX) systems. Further GPFS evolution includes:

- Starting with 1998, the new General Parallel File System – GPFS V1.1 – was introduced for high performance computing running on IBM RS/6000 SP using the IBM SP Switch as a high speed interconnect network and IBM's Virtual Shared Disk (VSD) as a storage abstraction layer.
- In 1999-2000, GPFS V1.3 evolved to address a wider set of applications running on SP systems, and was also used for the IBM Virtual Tape Server.
- In 2000, GPFS V1.4 added HACMP cluster support using SSA shared disks;
- In 2001, GPFS V1.1, the first version for Linux, became available, supporting up to 32 nodes in an IBM @server® Cluster 1300.
- Later in 2001, GPFS V1.5 for HACMP added support for the Fibre Channel attached Enterprise Storage Server® (direct attachment).
- In October 2002, GPFS 2.1 for AIX was released with a series of enhancements and new hardware support.

- In December 2002, an new release for GPFS on Linux (V1.3), adding new disk, networking, and updated Linux operating system support, as well as multiple functional enhancements. This version of GPFS also added support for the IBM @server Cluster 1350 managed by IBM Cluster System Management (CSM) V1.3 for Linux.
- In December 2003, a major announcement came out supporting GPFS V2.2 on IBM @server xSeries® Linux, pSeries Linux and pSeries AIX.
- Starting third quarter of 2004, GPFS V2.2 provides AIX-Linux inter operability for the three platforms mentioned above.
- A new release of GPFS - V2.3, with various performance and usability enhancements, new hardware support, and new configuration options is available since December, 2004.

Based upon the history, major releases come out at nearly one year interval, while minor releases coming throughout the year. At the time this redbook is written the latest minor release published is GPFS V2.3.0.5, while V2.3 is the major release version.

In Figure 2-1 presents a diagram of the GPFS evolution.

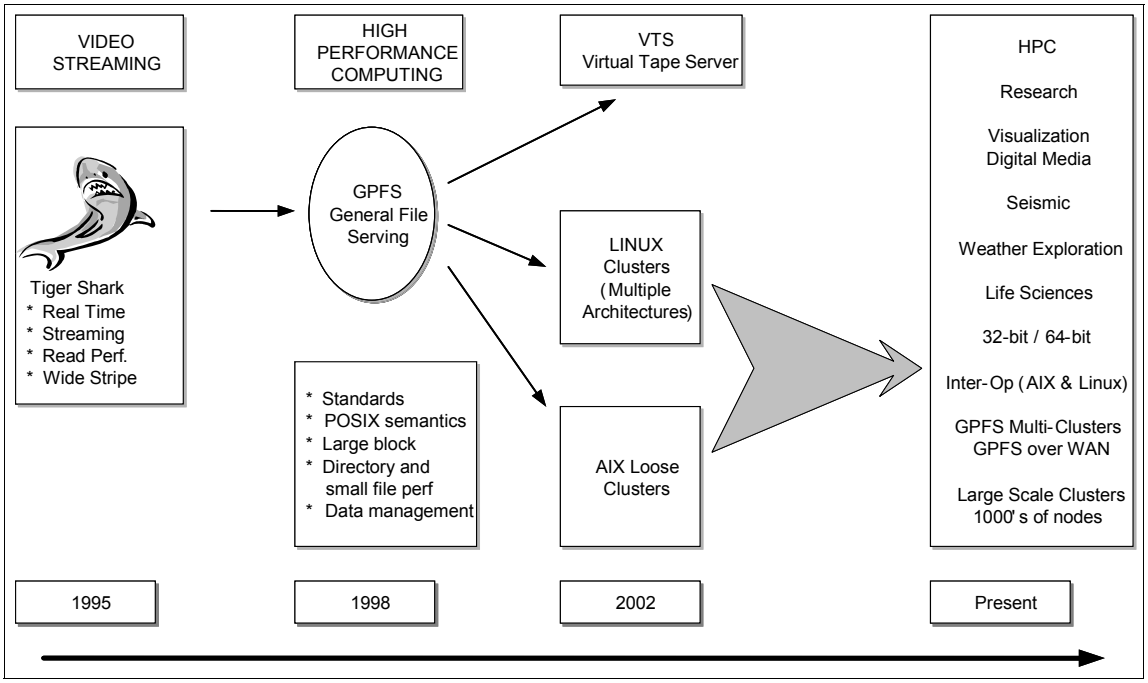

Figure 2-1 History of GPFS

2.1.1 Why choose GPFS?

This section presents some of the GPFS characteristics that should be considered in a decision making process when choosing which storage virtualization method (application) to use for your environment.

Note: The characteristics and facts presented here are may be subject to change (specially the numbers), therefore you should always check the latest GPFS information at:

http://www.ibm.com/servers/eserver/clusters/software/gpfs.html

- GPFS is highly scalable: (2000+ nodes)
  - Symmetric, scalable software architecture
  - Distributed metadata management
  - Allows for incremental scaling of system (nodes, disk space) with ease
- GPFS is high performance file system
  - Large block size (tunable) support with wide striping (across nodes and disks)
  - Parallel access to files from multiple nodes
  - Thorough token refinement (sector range) and token management
  - Efficient deep prefetching: read ahead, write behind
  - Recognize access patterns (adaptable mechanism)
  - Highly multithreaded daemon
  - Data shipping mode for MPI-IO and other applications
- GPFS is highly available and fault tolerant
  - Data protection mechanisms include journaling, replication, mirroring, shadowing (these are standard file system techniques)
  - Heartbeat mechanism to recover from multiple disk, node, connectivity failures
  - Recovery software mechanisms implemented in all layers

GPFS is in fact transparent to most applications, therefore virtually any applications can work with GPFS as though they were using a local file system. There are some restrictions, though, which must be understood to make sure that your application is able to deliver the expected results when using GPFS (application concurrent mechanisms, application locking characteristics, etc.). For details about the GPFS locking and recovery mechanisms see 2.2, “GPFS architecture overview” on page 35.

2.2 GPFS architecture overview

This section describes the GPFS architecture and the internal mechanisms that are used for various GPFS functions. The following is a short list of the GPFS components and functions:

- Components of GPFS (kernel extension, daemon, cache,...)
- GPFS management functions: Configuration Manager (CfgMgr), File System Manager (FSMgr), Metanode (MN)
- GPFS recovery procedures (for the Metanode, FSMgr, CfgMgr)
- GPFS token management and byte range locking
- GPFS replication (disk failure groups)
- GPFS quorum rules

2.2.1 GPFS components

On each node in the cluster, GPFS consists of the following components:

- Administrative commands
- A kernel extension
- A multi-threaded daemon

> **Note:** In addition to this list, on Linux nodes running GPFS an open source portability layer is used. This portability layer is “isolating” the licensed core of the GPFS daemon from the (open source) Linux kernel.

The Figure 2-2 on page 36 depicts the GPFS components.

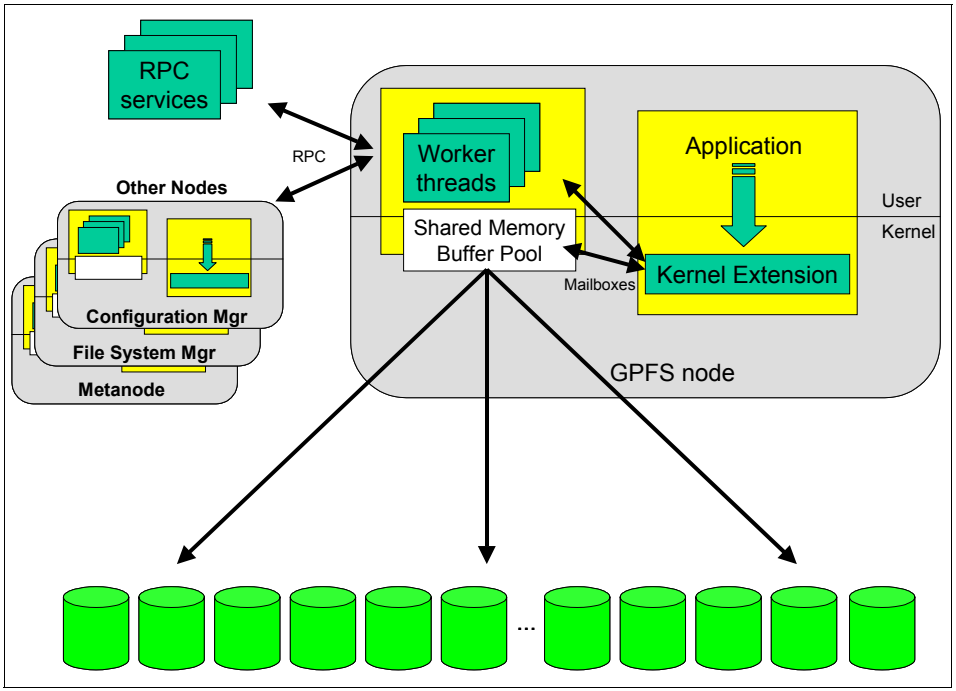

Figure 2-2 GPFS architecture

The multi-threaded GPFS daemon implements the management functions required for GPFS to function properly. The following section presents the GPFS management components.

2.2.2 GPFS file system management functions

There are three main file system related functions (roles):

- **Configuration manager** (CfgMgr)
  There is one node to assume this role per cluster; it is elected out of the pool of quorum nodes. The CfgMgr performs the following tasks:
  - Selects File System Manager node (out of the pool of manager nodes)
  - Determines if quorum rule is fulfilled
  - Initiates and controls the recovery procedures from node failure
- **File system manager** (FSMgr)
  There is one per mounted file system; this is assigned by the CfgMgr, and performs the following tasks:
  - Maintains the file system configuration
  - Manages disk space allocation

- Performs the token management (this thread runs only on FSMgr)
- Handles the quota management
- Provides the security services

- **Metanode** (MN)
  There is one metanode per open file; usually is the first node that has opened the file, and performs the following tasks:
  - Is the only node physically able to modify a file's metadata, therefore maintaining the file metadata integrity
  - Performs all metadata updates

Figure 2-3 shows the three management functions and their dependencies.

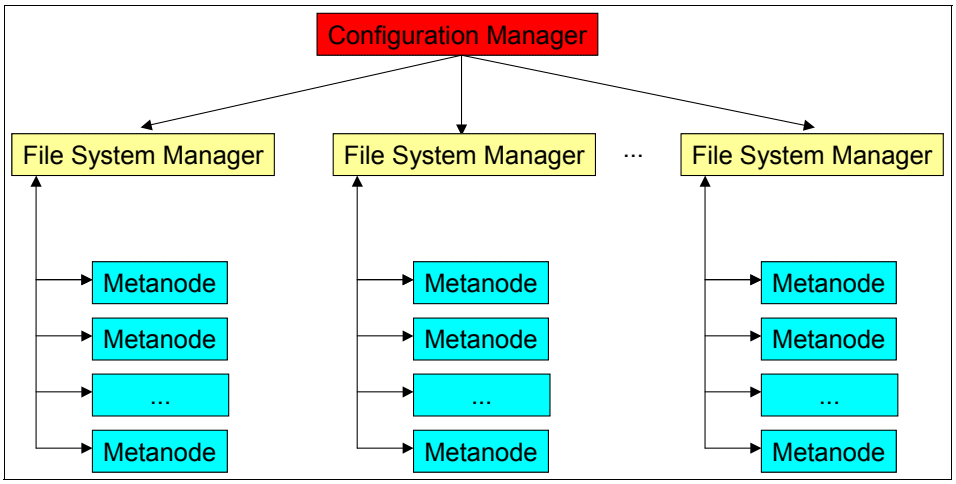

Figure 2-3 GPFS management functions and their dependencies

Note: The CfgMgr should not be confused with the Configuration Data Server. The Configuration Data Servers (primary and secondary) are the nodes, assigned at cluster creation (**`mmcrcluster`**) to maintain the GPFS configuration files (in /var directory). The primary and secondary nodes can be changed with **`mmchcluster`** command.

The CfgMgr is elected at GPFS cluster startup and is responsible for file system operations, like electing the file system managers. The file system managers (FSMgr) are elected out of the pool of quorum nodes during GPFS cluster initialization. Once assigned, the CfgMgr node CANNOT be changed (it will be automatically released from the remaining nodes in case of failure). The FSMgr can be changed by using the **`mmchmgr`** command (this is useful in case of maintenance operations).

2.2.3 GPFS block allocation

GPFS uses a typical file system logical structure, common in the UNIX space. A file system is essentially composed of a storage structure (data and metadata) and a file system device driver. It is in fact a very primitive data base used to store blocks of data and keeping track of their location and the available space.

The file system can be journaled (all metadata transactions are logged into a file system journal) for providing data consistency in case of system failure. GPFS is a journaled file system with in-line logs (explained further in this section). Figure 2-4 gives an overview of the file system logical structure.

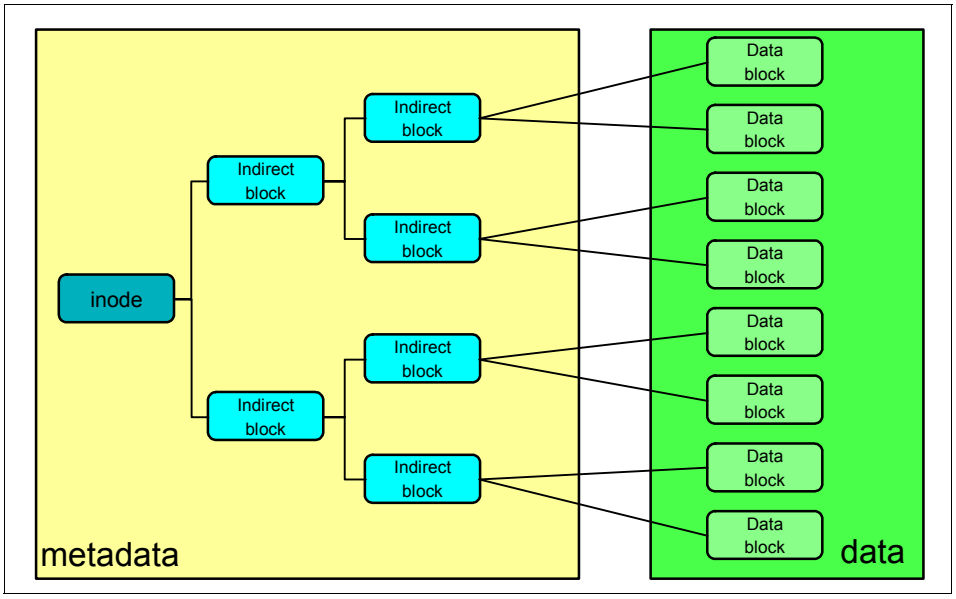

Figure 2-4 File system logical structure common in the unix space

The inodes and indirect blocks represent pointers to the actual blocks of data. For a journaled file system, all inodes and indirect blocks subject to a modification are stored into the file system log before they are actually modified. In addition to the inodes and indirect blocks, which keep track where the actual data is stored, the file system also keeps information on the available space by using a block allocation map.

The block allocation map is a collection of bits that represent the availability of disk space within the disks of the file system, which is maintained by the FSMgr. One unit in the allocation map represents a sub-block, or 1/32 of the block size of the GPFS file system. The allocation map is broken into regions which reside on disk sector boundaries.

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

Note: To avoid undesired locking contention, it is important to plan in advance your environment and have a good idea on how many nodes will actually access the file system.

Figure 2-5 shows the dependencies between file, GPFS block size, LUN and array segment size, and the physical to logical relation.

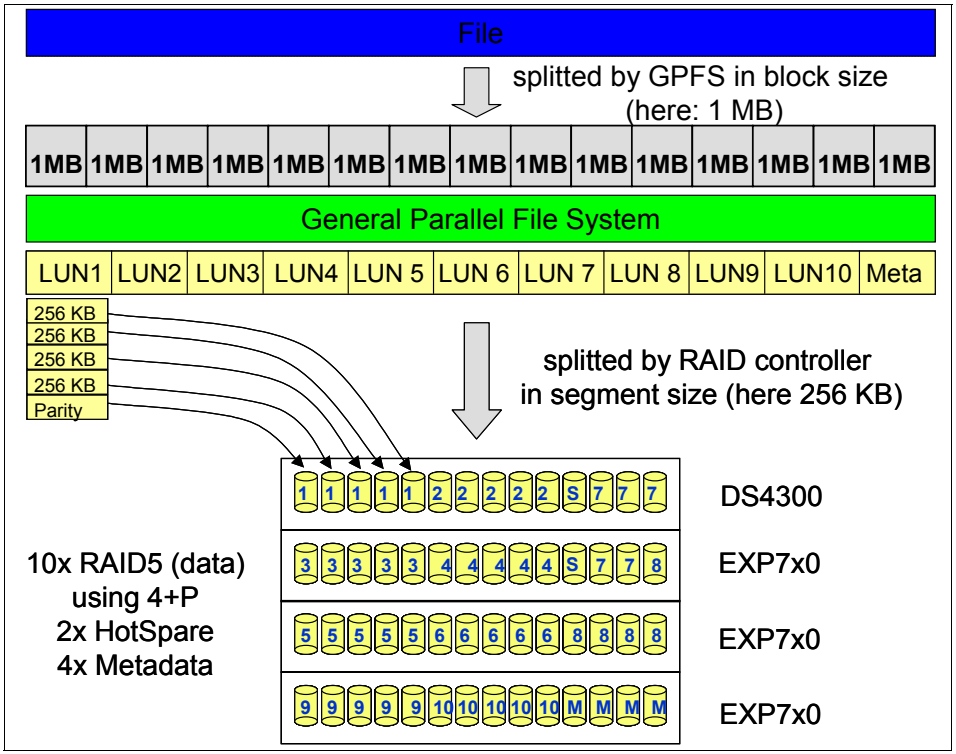

Figure 2-5 Dependencies between file, GPFS block size, LUN and segment size

2.2.4 Token management and byte range locking

The file system manager allocates blocks of disk space in ranges, using a token management mechanism. This means that a task can access a byte range within

a file (read or write) if it holds a token for that byte range. The token management function is distributed between two components:

- Token management server (i.e, “token server”)
- Token client

The token management server manages tokens on behalf of a particular file system. Therefore it resides on a node that acts as file system manager (FsMgr) for this file system. The token server distributes tokens to requesting token clients, and also distributes the lists of nodes to token clients requesting conflicting tokens. All requests are processed via the GPFS daemon (using threads).

There is one token client per node per file system running on behalf of all application tasks running on that node. But token operations are transparent to the application programmer, so any application can, and will use it without changing the application code.

The token client requests/holds/releases tokens on behalf of the task (application) accessing the file. So the task can access a given byte range without further access to the token manager once it has received a token, until another task attempts to access the same byte range.

In case of conflicting tokens, the task requests other task(s) holding conflicting tokens to release their tokens. The other tasks release their tokens to the token manager, but it will not do this until they have released the byte range lock for the file (this may include waiting for an I/O operation to complete). This means, that while accessing overlapping byte ranges where a file is being modified, the file operations will be sequentialized.

Figure 2-6 on page 41 shows an example of the token request procedure.

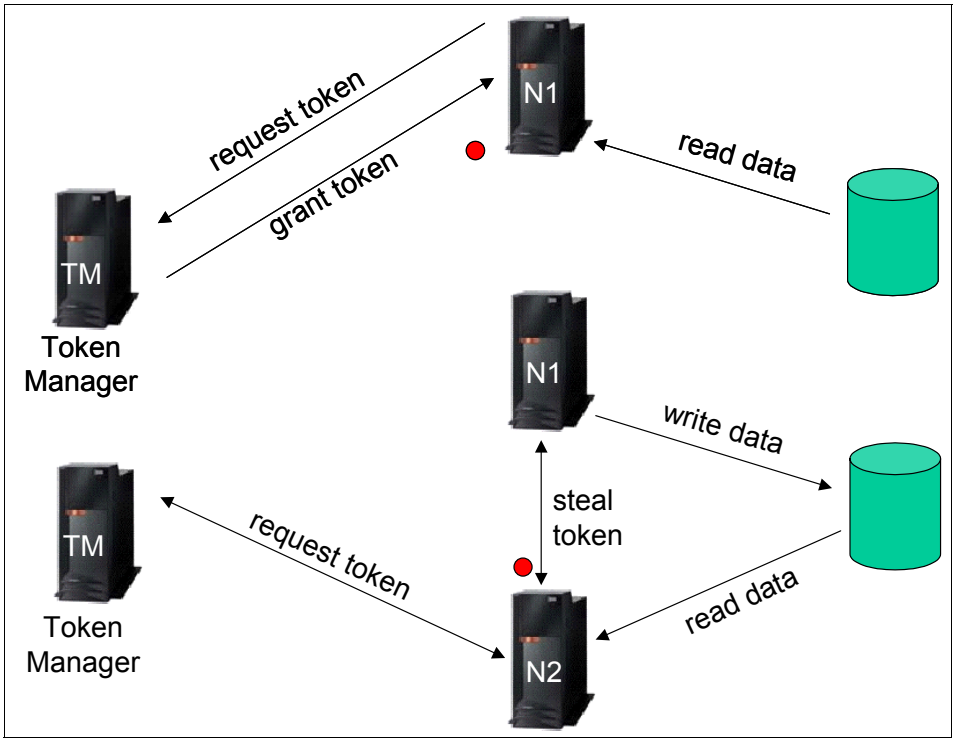

Figure 2-6 Token Management

First, a lock request for an object requires a message from node N1 to the token manager to obtain a token. The token server will grant a token to node N1 because node N1 is the first requesting access to the file. From now on, subsequent lock requests can be granted locally and data can be cached as long as a token is held.

If node N2 is requesting also a token for the same file as node N1 from the token server, a lock conflict occurs. The token server detects that a conflicting lock request is issued, and the token from node N1 is revoked. In case node N1 has write access to the file, modified data will be written to disk before the token is revoked. Node N2 will get the token, after node N1 has finished all required steps to revoke the token (See Figure 2-7 on page 42 for a locking conflict example).

Depending on the operation initiated, the process will choose different lock flavors. It will take whole file locking for less frequent operations (e.g., create, truncate, etc.), while finer grain locking will be taken for normal read/write operations.

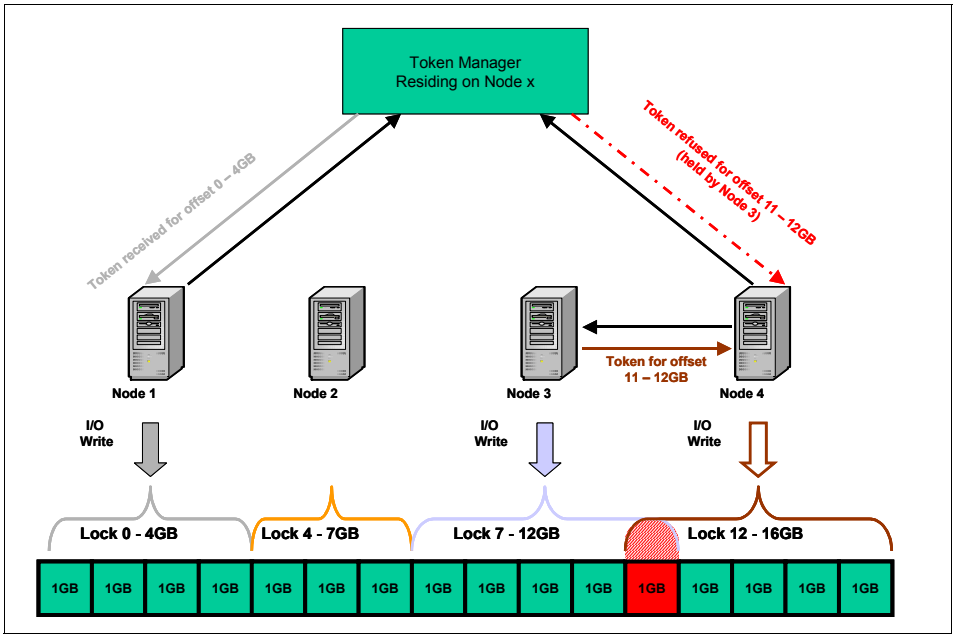

Figure 2-7 GPFS locking "conflict" example

The token manager (TM) manages all the tokens for byte-ranges (rounded to GPFS block boundaries), not the metanode. The metanode's only job is to hold the accurate metadata information for the file, i.e., the current inode and indirect block contents. It is the only node that can write this (meta)data to disk. All other nodes using the file get copies of the inode and indirect block data from the metanode after obtaining the necessary tokens.

When a Node 1 asks the TM for a byte-range that overlaps the range held by other nodes (2, 3, 4), the TM tells it which nodes have overlaps, and the requesting node asks the other nodes to relinquish the overlapping range and waits until the other nodes reply after any active operations in progress finish. The requesting Node 1 then tells then to the TM what ranges were relinquished by the other nodes and re-requests the range it needs. The TM updates its byte-range tables to reflect what was relinquished, and grants the requested range to Node 1.

Note: Technically, GPFS does not do "*whole file locking*" using byte ranges. For `truncate` and `setattr` operations (**`chmod/touch/chown`**), it gets an exclusive inode token (not byte-range) that forces all other nodes to write any dirty data, throw away any cached data, and relinquish their inode and byte-range tokens.

2.2.5 GPFS management functions availability

In addition to the token requests and the granularity of byte range locking described in the previous section, another mechanism is needed to check against failure of components (network adapters, disks, nodes — hardware and software). This instance checks the conditions of all nodes within the cluster, which is done within GPFS by the configuration manager (CfgMgr). The CfgMgr uses two different methods for this purpose:

- Disk leasing (request from the node to the CfgMgr)
- Pings (request from the CfgMgr to the node)

Both methods are using IP communication to check if the nodes within the cluster are alive and responsive. Under normal operations the only messages exchanged between the CfgMgr and the other nodes are the lease renewals.

Pings are sent only when a node fails to renew its lease in a timely manner and the CfgMgr suspects the node may have failed. The purpose of the pings is to prevent “false downs”, i.e., declaring a node down that is still up, but heavily loaded. This leads to three different situations the configuration manager has to investigate:

1. Normal operation (diskLeasing only) (see Figure 2-8 on page 44)
2. Node fails both diskLease renewal and responding to pings (see Figure 2-9 on page 45)
3. Node fails to renew its diskLease, but responds to pings (see Figure 2-10 on page 46)

Before we explain the three cases in detail, we first need to understand the parameters involved in the decision about a failure of a node, which are:

- **leaseDuration**
  Time a disk-lease is granted from CfgMgr to any node.
  Default: 35 seconds
- **leaseRecoveryWait**
  Additional time to wait in case a node failed to renew the lease. The purpose allows for I/Os that were started just before the lease ran out to finish first.
  Default: 35 seconds
- **PingPeriod**
  Time period between two pings (second)
  Default: 2 seconds
- **totalPingTimeout**
  Total amount of time pinging a node before giving up (second)
  Default: 120 seconds

- **minMissedPingTimeout**
  Minimum amount of time pinging a node without a reply (second)
  Default: 8 seconds
- **maxMissedPingTimeout**
  Maximum amount of time pinging a node without a reply (second)
  Default: 60 seconds

Case 1 - normal operation (disk leasing only)

In normal situations, every GPFS node requests a lease from the CfgMgr. After the leaseDuration expires, every GPFS node needs to renew its lease. This is shown in Figure 2-8.

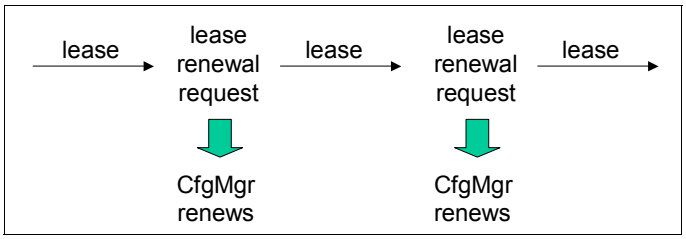

Figure 2-8 GPFS disk leasing - normal operation

Case 2 - Node fails both (diskLease, and pings)

In case a node misses both, the renewal of its disk lease and to answer to the pings, the CfgMgr will declare this node as suspect. Now we have to differentiate again between two scenarios:

1. The GPFS daemon has died, but the node is still up. In this case the node is declared dead immediately, because the operating system closes the TCP connection to the CfgMgr, which will be detected by the GPFS daemon on the CfgMgr node immediately.
2. The node is really dead or disconnected (e.g., power turned off or missing network link). In this case the node will not reply to pings and will be declared dead after a time-out, equal to the *(leaseDuration + leaseRecoveryWait)* seconds after it last renewed its lease.

In both cases, file system log (journal) recovery still has to wait until the node's lease has expired plus *leaseRecoveryWait* seconds before it starts. This is shown in Figure 2-9 on page 45.

Note: In the first case, even though the node is declared dead, file system log (journal) recovery still has to wait until the nodes lease has expired plus leaseRecoveryWait seconds before it starts. Hence in both first and second cases, log recovery will begin *(leaseDuration + leaseRecoveryWait)* seconds after the node last renewed its lease.

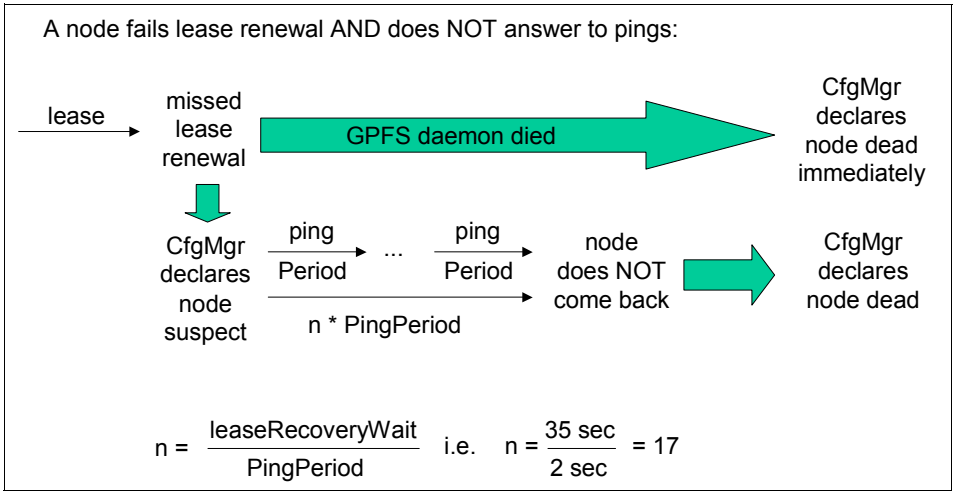

Figure 2-9 GPFS failure case - disk renewal and pings missed

Case 3 - Node fails to renew its diskLease, but responds to pings

The node is “catatonic”, i.e., the node still replies to pings, and it has not closed its TCP connection to the CfgMgr node, but somehow does not renew its lease. Normally this state is an indication that either the system is severely overloaded, or the operating system is “hung”, where processes no longer run, but network interrupts are still being serviced.

In this case, the node will be declared dead after a time-out, which is *leaseDuration + totalPingTimeout* seconds after it last renewed its lease and initiates a node recovery procedure for the failed node. This is shown in Figure 2-10 on page 46.

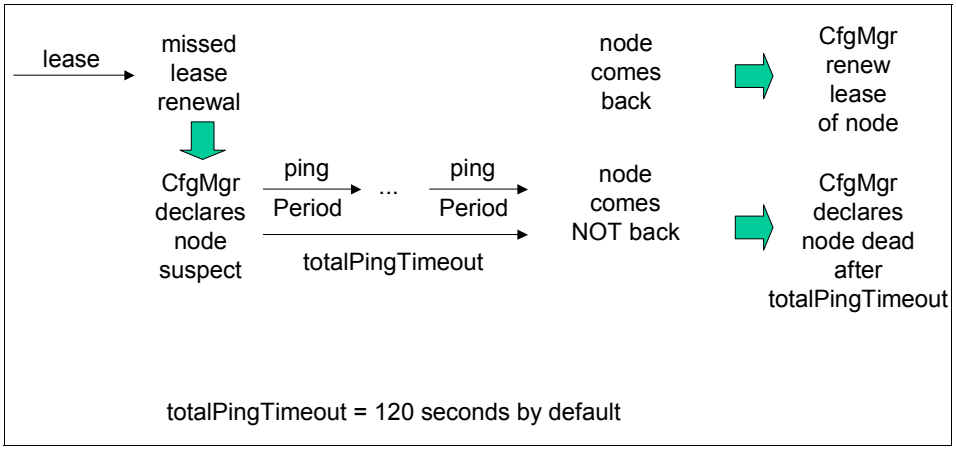

Figure 2-10 Node disk lease renewal missed, but still answers to pings

Note: In this third case, log recovery still has to wait for *totalPingTimeout*. Hence, log recovery will begin *(leaseDuration + totalPingTimeout)* seconds after the node last renewed its lease.

Table 2-1 Node operation comparison showing the detection, and the log recovery starting time

Case	Description	node detected dead	time (seconds) using defaults	log recovery starts	time (seconds) using defaults
1	normal	n/a	n/a	n/a	n/a
2	gpfs daemon died	immediately	0	(leaseDuration + leaseRecoveryWait) seconds after the node last renewed its lease	~35 - 70
	node dead, or disconnected	(leaseDuration + leaseRecoveryWait) seconds after it last renewed its lease.	~35-70	(leaseDuration + leaseRecoveryWait) seconds after the node last renewed its lease	~35 - 70
3	catatonic	(leaseDuration + totalPingTimeout) seconds after it last renewed its lease.	~120-155	(leaseDuration + totalPingTimeout) seconds after it last renewed its lease.	~120 - 155

Note: The time intervals shown in the previous table are calculated based on the assumption that a single event occurs at a time. If multiple events occur, they will be processed in sequence, therefore the cluster and/or file system recovery times may vary.

The calculation is valid for GPFS using the disk leasing mechanism, as the hardware reservation mechanism is not a valid option starting GPFS V2.3.

2.2.6 GPFS internal recovery procedures

In this section we want to answer the following question: How does a failure of a GPFS management function (CfgMgr, FSMgr, MN) affect I/O operations?

For the three management functions, we are investigating the following three scenarios:

1. Metanode (MN) fails, but this node does not act as CfgMgr or FSMgr node
2. File system manager (FSMgr) fails, but this node does not act as CfgMgr
3. Configuration manager (CfgMgr) fails

In case a node handles multiple management functions, (for example, is a metanode and FSMgr), all rules of both conditions take place.

Attention: Only one failure (state change), such as the loss or initialization of a node, can be processed at a time, and subsequent changes will be queued. This means that the entire failure processing must complete before the failed node can join the group again. All failures are processed first, regardless the type of failure, which means that GPFS will handle all failures prior to completing any recovery.

Scenario 1: Metanode fails

In case the MN fails, the CfgMgr invokes recovery for each of the file systems that were mounted on the failed node. The recovery is done by the FSMgr for each file system (mounted on the failing node) to ensure the failed node no longer has access to the disks belonging to those file systems.

The recovery will rebuild the metadata that was being modified at the time of the failure to a consistent state with the possible exception that blocks that might have been allocated but are not part of any file and are effectively lost until a file system check (**`mmfsck`**) is run, either online or offline.

Note: Whenever possible, we recommend to run the **mmfsck** with the file system unmounted on all nodes (offline). Running a file system check offline allows for recovering from errors which may not be possible to correct when the file system is mounted.

After file system log recovery is complete, the locks held by the failed nodes are released for this file system. Completion of this activity for all file systems completes the failure processing. Only the completion of the protocol allows a failed node to rejoin the cluster.

This means that the tokens given for the files the MN owned before it failed are frozen. Therefore a read or write access can proceed under the assumption that the surviving node has a token “stronger” than the ones owned by the failing MN, and the write operations do not need to be committed immediately.

The metadata updates are cached and will be sent to the metanode the next time the sync daemon wakes up. But writes that need to be committed immediately (e.g., appending to the end of the file), or synchronous writes cannot proceed further, and will be interrupted until the node recovery procedure is finished and a new metanode is assigned.

Attention: GPFS (like any high level clustering products involving external storage) provides a takeover mechanism for the managed storage, but is not capable to perform takeover instantly, because of numerous other uncontrollable operating system and disk I/O activities and events that take unknown (estimated) amounts of time (this is not controllable by GPFS). To avoid failover time, a mechanism outside GPFS may be needed to avoid application interruption.

Scenario 2: File system manager (FSMgr) fails

If FSMgr node fails, operations continue normally elsewhere using existing cached tokens, but some operations will wait until there is a new FSMgr is assigned (for example, if a node needs to acquire tokens for files or directories that are not cached). In this case, file system operations (like read or write) could be stopped immediately or after some time, depending on the cached data state.

To replace the failed FSMgr, a new FSMgr will be assigned by the CfgMgr, which chooses a node out of the pool of management nodes. If the file system manager is newly appointed as a result of this failure, it rebuilds token state by querying the other nodes of the cluster. After rebuilding the token state is complete, the log recovery (journal recovery) for the failed node proceeds.

There are situations where it may be impossible to appoint a new file system manager. Such situations involve the failure of paths to disk resources from multiple (if not all) nodes. In this event, the configuration manager nominates several nodes (by their host names) to successively try to become the file system manager. If none is successful, the configuration manager unmounts the file system everywhere.

Scenario 3: Configuration manager (CfgMgr) fails

A new CfgMgr is chosen through an election process held between the set of designated quorum nodes. Operations proceed normally until the lease needs to be renewed or a node configuration change is requested. In this case, the operations will be suspended until the new CfgMgr is elected.

Note: The Configuration Manager is not to be confused with the Configuration Data Server. The Configuration Data Servers (primary and secondary) are the nodes, assigned at cluster creation (**`mmcrcluster`**) to maintain the GPFS configuration files. The Configuration Data servers can be changed with **`mmchcluster`** command.

The Configuration Manager is elected at GPFS cluster startup and is responsible for file system operations, like electing the file system managers. The file system managers will be elected out of the *pool of quorum nodes* during GPFS cluster initialization. The chosen node can be changed by using the **`mmchmgr`** command.

In case of a node failure incarnating multiple “personalities” (management functions), the previously described three scenarios combine the actions to perform recovery.

Figure 2-11 on page 50 depicts an overview of the GPFS internal operation.

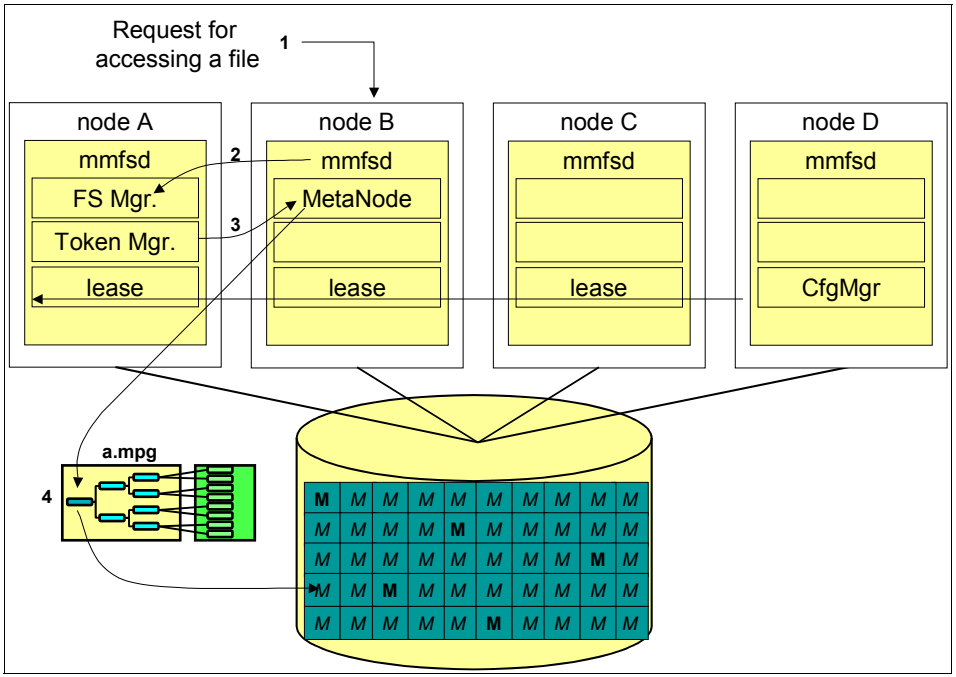

Figure 2-11 Relationship and workflow between components, functions, and data access

2.2.7 Replication (disk failure groups)

As we have mentioned before, GPFS is a journaling file system (logs all metadata activity). To maintain the file system online (mounted) and consistent, several protection mechanisms are used:

- Data and metadata replication
- Log file replication
- Disk failure groups

The disk failure groups are used for keeping the file system (not the cluster!) online based on maintaining a quorum file system descriptors (disks) available. If you are familiar with AIX logical volume manager, the information about a volume group is stored on the Volume Group Descriptor Area (VGDA), and a quorum of VGDAs must be available for the VG to be either brought online or to remain varied on.

GPFS reserves a storage area on every disk which is part of a file system for the file system descriptor (FSDesc), but by default writes the FSDesc on three of the disks which are part of that file system (if available).

This behavior poses the problem that in case of multiple disks failure, the file system may be unmounted due to the lack of FSDesc quorum (even though the total number of disks still available for that file system is larger that 50%+1).

To solve this problem, GPFS uses disk failure groups. The characteristics of the failure groups are:

- A number identifying the failure group to which a disk belongs
- Is used by GPFS during metadata and data placement on the disks of a file system
- Ensures that no two replicas of the same block will become unavailable due to a single failure
- Are set by GPFS by default
- Can also be defined either at NSD creation time (**mmcrnsd**/**mmcrvsd** commands) or later (**mmchdisk** command)
- The syntax for the disk descriptor file is the following

  ```
  DiskName:PrimaryNSDServer:BackupNSDServer:DiskUsage:FailureGroup:NSDName
  ```

- It is important to set failure groups correctly to have proper/effective file system replication (metadata and/or data)

The following paragraphs explain in more detail considerations for choosing the hardware and the failure groups to provide maximum availability for your file system.

Data replication

The GPFS replication feature allows you to specify how many copies of a file to maintain. File system replication assures that the latest updates to critical data are preserved in the event of hardware failure. During configuration, you assign a replication factor to indicate the total number of copies you want to store, currently only two copies are supported.

Replication allows you to set different levels of protection for each file or one level for an entire file system. Since replication uses additional disk capacity and stresses the available write throughput, you might want to consider replicating only file systems that are frequently read from but seldom written to or only for metadata. In addition, only the primary copy is used for reading, unlike for instance in RAID 1, hence no read performance increase can be achieved.

Failure groups

GPFS failover support allows you to organize your hardware into a number of failure groups to minimize single points of failure. A failure group is a set of disks that share a common point of failure that could cause them all to become simultaneously unavailable, such as the disk enclosures, RAID controllers or storage units such as the DS4500.

In order to assure file availability, GPFS maintains each instance of replicated data on disks in different failure groups. This is significantly different from replication that can be performed by some storage subsystems, such as the DS4500. See Figure 2-12 and Figure 2-13 on page 53. Even if no replication is specified when creating a file system, GPFS automatically replicates recovery logs in separate failure groups.

In case of a hardware failure different things happen depending on the configuration:

- **Disk failure**
  is handled by the disk controller depending on the RAID configuration.
- **LUN failure**
  is handled by GPFS, who redirects I/O operations to the secondary copy of the file where the primary I/O block is effected and where the secondary copy is effected no copy is written for write I/O requests. When the LUN has been recovered, a manual intervention is needed in order to ensure the consistency of the two data copies held by GPFS.

Example 1 (see Figure 2-12):
In this configuration the DS4500 replicates all its LUNs to a second DS4500, using the back-end loops transparently to GPFS. This solution provides protection against loss of data, if you lose your DS4500#1 containing the primary copies of the file, indicated by number and solid color, a second copy will be made available and GPFS will use the secondary copies, indicated by number' and shaded color, maintained on DS4500#2 without being aware of the failure. See Figure 2-12, where the I/O blocks are written across the LUNs, residing on DS4500#1, the controller copies the changes to the LUN. This can introduce slight performance improvement on writes if write cache is enabled. However for reads the performance slows down as only DS4500#1, containing the primary copy is used.

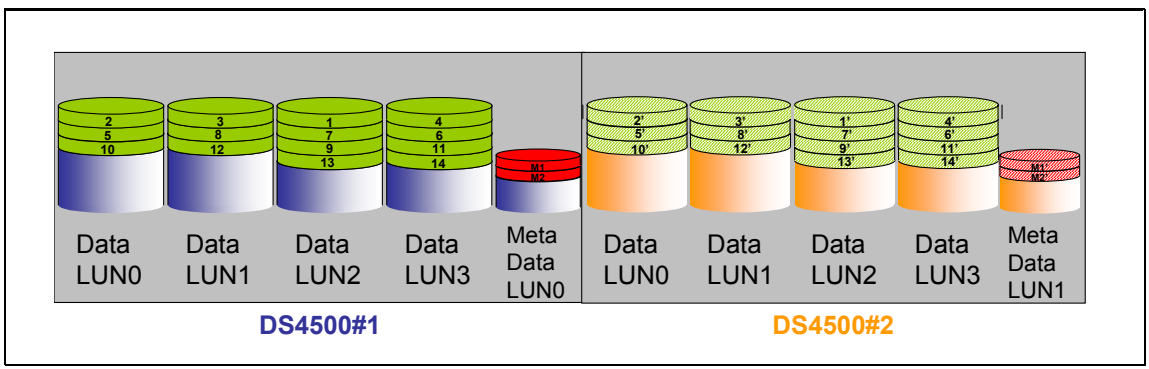

Figure 2-12 Mirroring between two DS4500

Example 2 (Figure 2-13):
In this configuration the LUNs configured by GPFS into two failure groups, with the DS4500 being the common point of failure, hence each DS4500 forms it's own failure group. The write launches two Round Robin algorithms that independently choose the LUN they are going to use for the I/O block, but monitor each other in order to ensure that if the first copy of the I/O block gets written into the LUN in failure group 1 then the second copy gets written in failure group 2. This improves the reading performance, which is distributed between the two storage enclosures, as GPFS only uses the primary copy for reading.

As shown in Figure 2-13, the first write I/O block is written on Data LUN 5, residing in failure group 2, and the second copy of the same I/O block is written on Data LUN 2, in failure group 1.

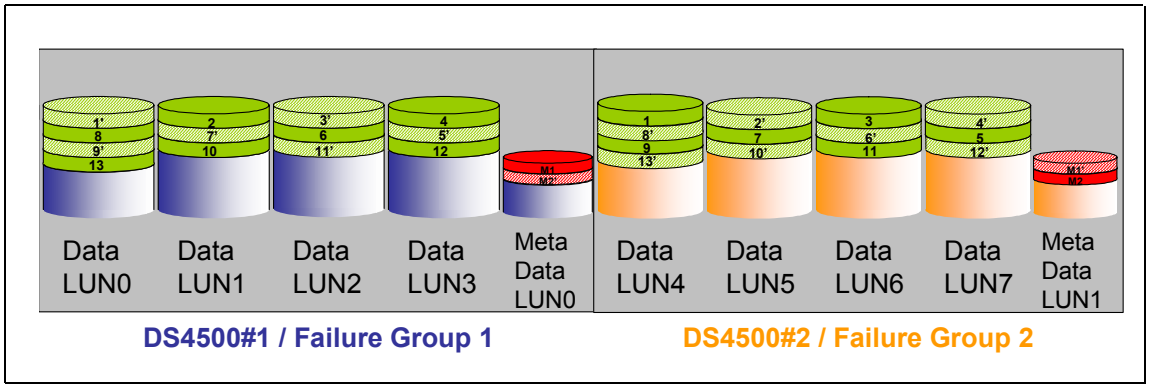

Figure 2-13 GPFS replication using two DS4500

GPFS is unaware of the physical implementation of the LUNs. So, it's up to the underlying technology of the storage system to guarantee the availability of each LUN. However, if the files are striped over LUNs belonging to different storage systems, then the loss of a storage system would compromise every data file of the file system.

No single storage system can guarantee the availability of another independent storage system. One possible solution would be mirroring of storage systems amongst themselves (e.g., PPRC). This will however delay the point in time when the data is fully protected. Since this activity is not synchronized with the GPFS file system, it may add a certain degree of complexity to the storage architecture.

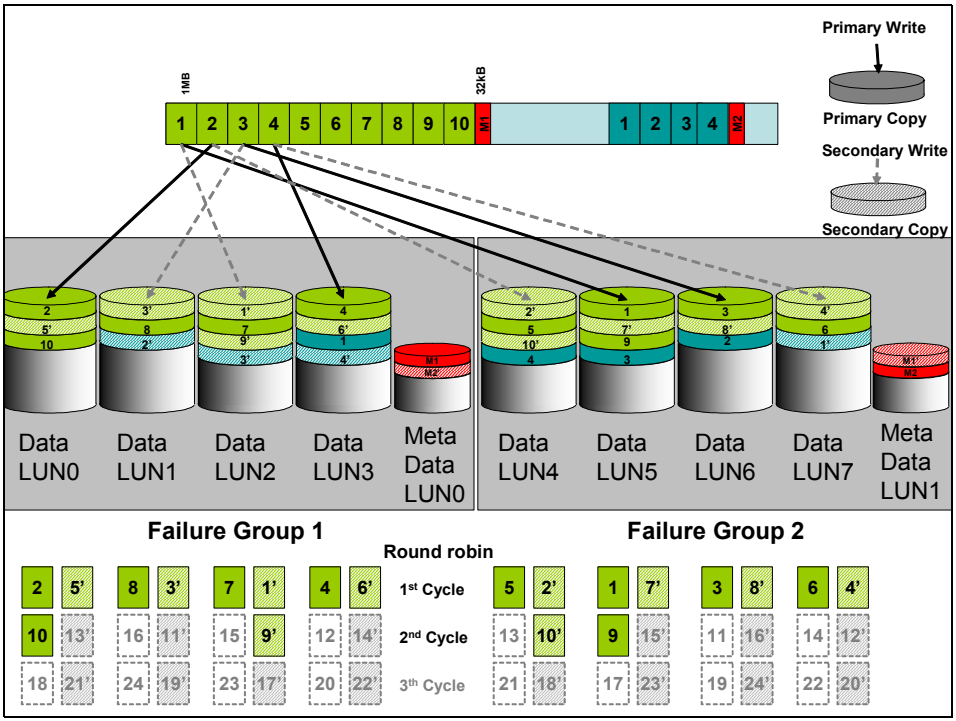

Figure 2-14 GPFS replication using Failure Groups

To overcome this issue, GPFS uses *failure groups* (see Figure 2-14). The failure groups are defined as storage entities that have a common point of failure. Therefore, a single storage system can be defined as a failure group. GPFS then mirrors the data over the different failure groups.

Each write request is executed towards each storage group, creating in effect 2 copies of the same file or even same I/O block, each copy residing on a different failure group. In case of a lost failure group, or even a lost LUN within a failure group, GPFS uses the data copy on the second failure group. It is therefore able to protect file striping over different independent storage systems, which virtually removes the scalability limits of the file system.

2.2.8 Quorum rules

There are two types of quorum: cluster quorum and file system quorum.

Cluster quorum is used to prevent a cluster becoming partitioned. Should this happen, then each partition would have a unique file system manager and the

state of the file system would become inconsistent. There are two different cluster quorum options implemented in GPFS:

- **Node quorum**
  It is defined as quorum minimum == *number_quorum_nodes/2 + 1*
  Only a subset of nodes are designated as quorum nodes. This speeds up cluster formation as well as allows any number of non-quorum nodes from failing without impacting the overall quorum calculation.
- **Disk tiebreaker quorum** (new in GPFS 2.3)
  In this case, GPFS uses only two nodes for quorum nodes and up to three disks (NSDs) directly attached to the two designated quorum nodes. The voting mechanism in this case ensures that one side of the cluster will remain alive based on the access to the shared disks. Imagine the start of a basketball game, when the referee throws the ball vertically, and one of the teams will have the ball possession regardless.
  - Can be used in configurations with two or more nodes
  - Odd number of disks is recommended (1, 3)
  - Disks (NSDs) must be accessible from the quorum nodes
  - Recommended that these quorum nodes are also the primary and secondary GPFS configuration servers

  Use the `tiebreakerDisks` parameter on the **`mmchconfig`** command to enable this quorum mechanism:

  ```
  mmchconfig tiebreakerDisks="gpfs1nsd;gpfs2nsd;gpfs3nsd"
  ```

Note: There are special considerations in a two node cluster. To maintain the quorum, in previous GPFS version (2.2), a hardware reservation mechanism was used but only certain storage subsystems were supported (the list of storage subsystems supported in a two node cluster contained only a subset of the GPFS supported list).

In GPFS 2.3 with the new disk tiebreaker mechanism, virtually any storage supported by GPFS can be used in a two node cluster. This is possible due to the new disk tiebreaker mechanism (which requires two nodes anyway).

In addition to cluster quorum, for each file system to survive a quorum of descriptors must be available - file system configuration quorum:

- **File system descriptor quorum**
  A file system descriptor is distributed to three disks across disk failure groups. Two of these disks are needed to hold the file system active.

File system descriptor

For maintaining file system integrity and availability, GPFS uses a File System Descriptor (FSDesc). For this purpose, GPFS originally reserves space on all disks in assigned for a particular file system (at the file system creation time), but only writes the FSDesc to a subset of three disks (if available). As changes to the file system occur (such as adding or deleting disks), the FSDesc needs actually to be maintained on that subset of disks.

The FSDesc contains information about the disks which are part of that file system, and also time stamp information for the last file system modification that has occurred (adding/deleting disks or modifying file system structure).

In order to determine the correct file system configuration, a quorum of the special disk subset consisting of minimum three disks (containing the FSDesc), used as a file system configuration quorum, must be online, so that the most up-to-date FSDesc can be found. GPFS distributes the copies of FSDesc across disks and failure groups.

If you are using a single storage subsystem (which is generally not recommended for a production environment), all FSDesc copies will be in the same failure group, therefore it is rather difficult to predict the file system behavior in case of multiple LUN failure. If one disk (containing a copy of the FSDesc) fails, the FSDesc will be moved to a new disk (space for FSDesc is allocated at the file system creation time), and also the other two FSDesc copies are updated. However, if two of the three disks containing the FSDesc die simultaneously, the FSDesc copies cannot be updated to reflect the new quorum configuration and the file system must be unmounted to preserve existing data integrity.

If you are using only two failure groups, one failure group contains two copies of the FSDesc, and the other failure group has one copy. In a scenario that causes one entire failure group to fail, therefore rendering half of the disks unavailable, if the surviving contains two FSDesc copies, the file system remains active (see Figure 2-15 on page 57).

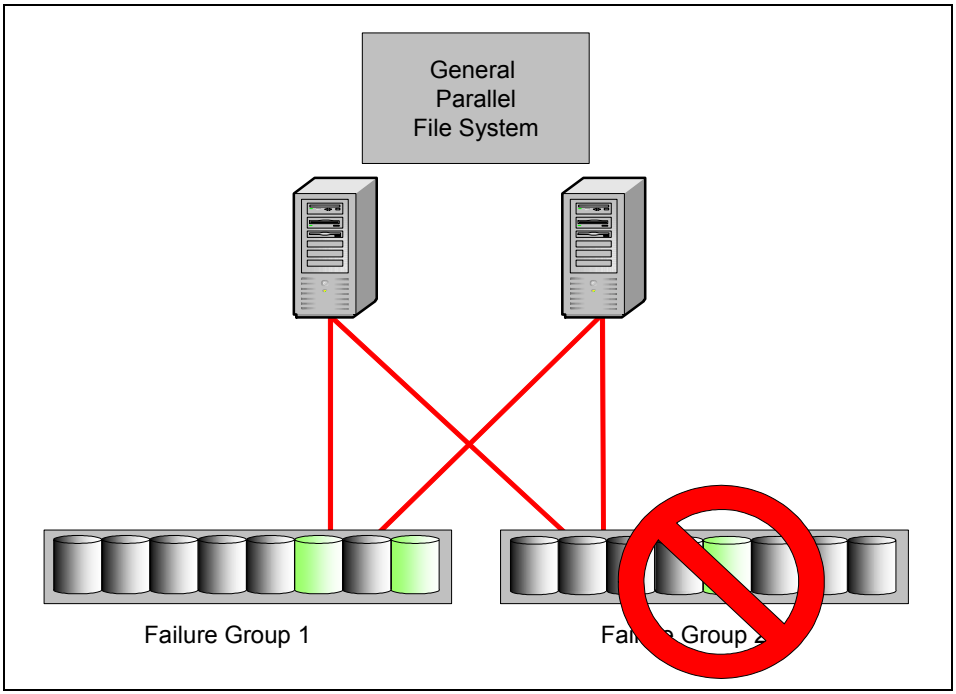

Figure 2-15 Failure group 2 with one file system descriptor fails

If the lost failure group contains a majority of the FSDesc quorum, the FSDesc cannot be updated and the file system will unmounted (forced) on all nodes. See Figure 2-16 on page 58.

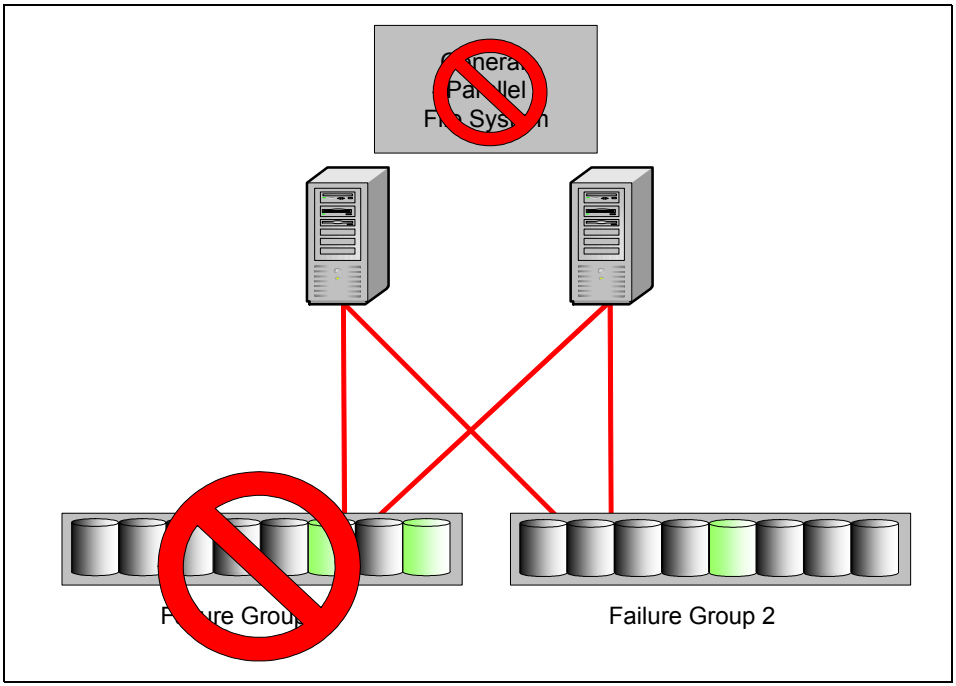

Figure 2-16 Failure group 2 with two copies of file system descriptor fails

Therefore, it's important to have a minimum of three failure groups in order for a file system to survive a loss of a single failure group (see Figure 2-17 on page 59).

One elegant way to maintain a file system alive is to designate a separate disk as a tie breaker. This disk (NSD) can be configured only as a FSDesc holder, there is no need to use this disk for storing data or metadata. In this case, this disk does not have to be in a directly attached fiber enclosure. Its speed and capacity are less important, and therefore can physically reside on an internal disk on a node within the cluster. If you are using a node quorum in your cluster it may be a good idea to designate this node as a quorum node also ("quorum buster").

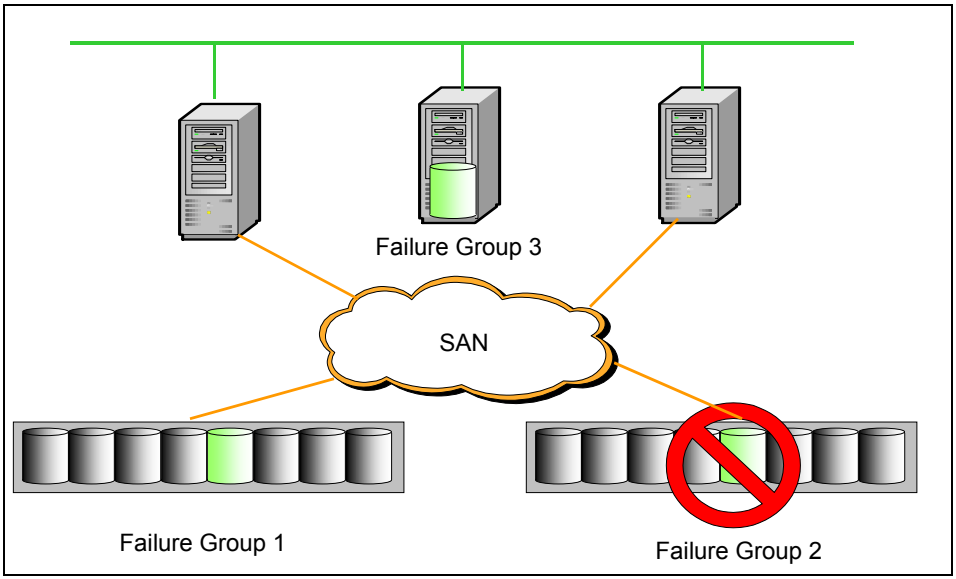

Figure 2-17 Three failure groups (each one holding a copy of the FSDesc)

As you can see in Figure 2-17, the third failure group is stored on a disk which is not even part of the SAN (not used for data or metadata storage). In this case, this failure group cab reside on a disk inside a node in a different location, therefore providing for file system survivability in case of a disaster recovery scenario.

2.2.9 GPFS memory management

This section presents an overview of the GPFS daemon memory usage, and is intended to emphasize how GPFS handles different operations.

GPFS uses three areas of memory:

- Memory allocated from the kernel heap
  The kernel memory is used for control structures such as v-nodes and related structures that establish the necessary relationship with Linux.
- Memory allocated within the daemon segment
  The file system manager node requires more daemon memory since token state for the entire file system is stored there. The daemon memory is used for structures that persist for the execution of a command or I/O operation, and also for states related to other nodes.
- Shared segments accessed from both the daemon and the kernel
  Shared segments consist of both pinned and unpinned storage, which is

allocated at daemon start-up. The pinned storage is called *pagepool*, and is controlled by configuration parameters. In a non-pinned area of the shared segment, GPFS keeps information about open and recently accessed files. This information is held in two forms:

– A full inode cache
  The inode cache contains copies of inodes for open files and for some recently used files which are no longer open.

– A stat cache
  The stat cache contains enough information to respond to inquiries about the file (such as **`ls -l`**), but not enough information to read from it or write to it. A stat cache entry consumes about 128 bytes which is significantly less memory than a full inode. GPFS will prefetch data for stat cache entries if a pattern of use indicates this will be productive. The stat cache entries are kept for:

  - Recently accessed files
  - Directories recently accessed by a number of **`stat()`** calls

Pagepool and malloc pool

The pagepool is used by the GPFS for caching data, and the malloc pool is used by the GPFS for caching inodes and metadata. Each inode occupies approximately 800 bytes from the malloc pool.

Caching allows GPFS to implement read ahead and write behind mechanisms (for hiding the latency of the disk storage system). GPFS is a multi-threaded daemon, this means that as soon as application's write buffer has been copied into the pagepool, the write request is complete from the application's perspective. GPFS then schedules a worker thread to manage the write request to completion by issuing I/O calls to the disk device driver (write behind). See Figure 2-18 on page 61. If the system is busy, this write can be delayed for up to (maximum) one minute, after this the thread gets priority and flushes the dirty data cache blocks to the disks. Metadata is flushed every 30 seconds in 512KB blocks.

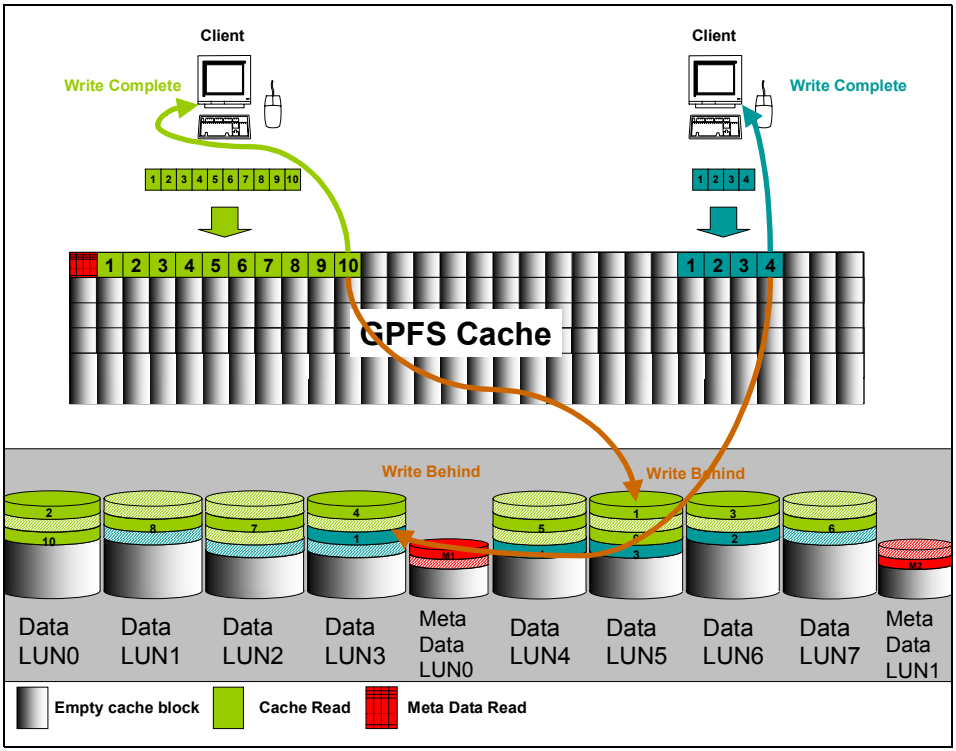

Figure 2-18 Write behind

GPFS computes a desired number of blocks to read ahead for each open file based on the performance of the disks and the rate at which the application is reading data (either sequentially or using pattern recognition). If an additional prefetch is needed, a message is sent to the daemon that will process this request asynchronously with the completion of the current read.

With some access patterns, increasing the amount of pagepool storage may increase I/O performance for file systems with the following operating characteristics:

- Heavy use of writes that can be overlapped with application execution
- Heavy re-use of files and sequential reads of a size such that pre-fetch will benefit the application, see Figure 2-19 on page 62.

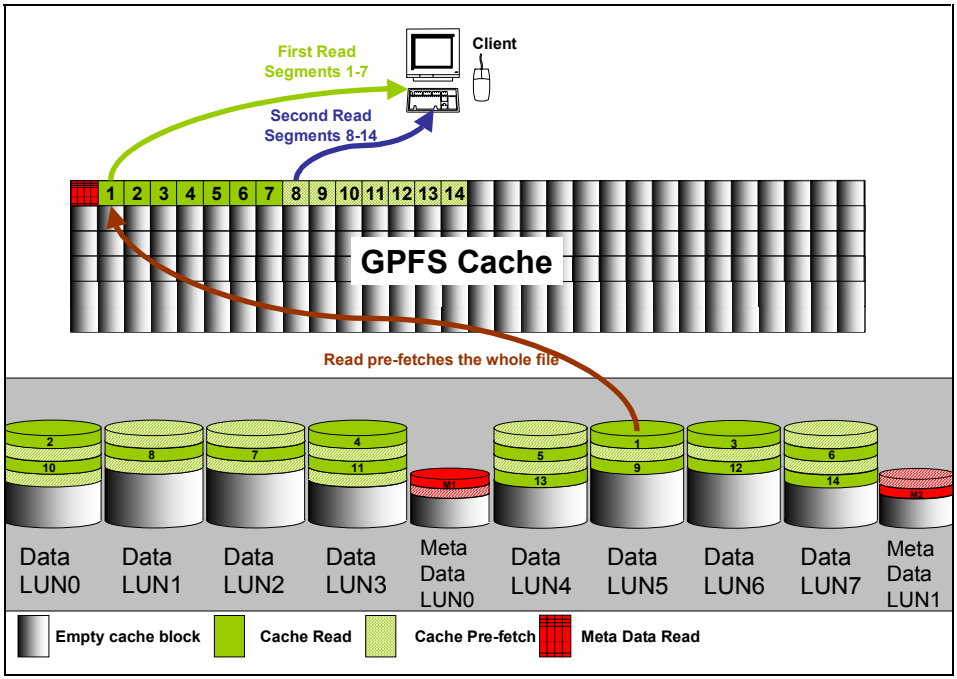

Figure 2-19 GPFS prefetch

2.2.10 GPFS cluster models

In the current GPFS V2.3, disk access is achieved via the Network Shared Disk (NSD) mechanism which is actually a way of providing raw block disk access access over an IP network. Unlike IBM's Virtual Shared Disk (VSD), this disk abstraction mechanism does not create the actual devices in `/dev` directory, rather keeps the access to the NSD devices for exclusive GPFS use.

The current GPFS implementation (V2.3) offers five cluster models (variations are possible also):

1. Dedicated NSD server(s) model
2. Direct disk attached model
3. Mixed model (combination of dedicated NSD servers and direct disk attached model)
4. Joined model (heterogeneous environment consisting of AIX and Linux hosts)
5. Cross-cluster model (file systems belonging to different clusters can be cross-mounted between clusters)

Note: There is no preferred model, as this is application and/or customer environment determine the appropriate design.

Each model has its limitations and its strengths, and each one is commonly used. For example, within digital media broadcasting environments, the direct attached model is the preferred solution. The following sections introduce the previously listed GPFS models. For additional information and variations of these models, see *General Parallel File System (GPFS) for Clusters: Concepts, Planning, and Installation*, GA22-7968.

Before explaining the different GPFS cluster models, it is essential to understand the differences between Linux and AIX regarding the network shared disk (NSD) layer implementation. Figure 2-20 shows all possible NSD options in a GPFS environment.

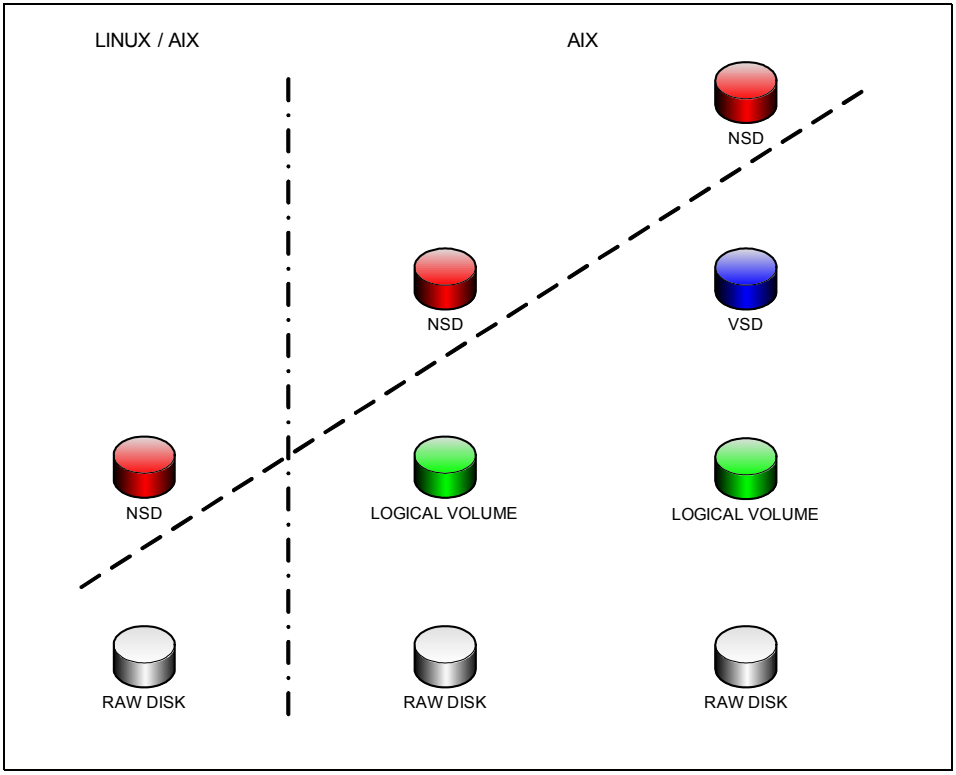

Figure 2-20 GPFS - understanding the NSD-layer

While GPFS on Linux only supports the use of the NSD layer on top of a raw disk (which is also preferred choice for AIX), in AIX GPFS still supports two other

choices. Option two is using NSD on top of the AIX logical volume manager (LVM). AIX also offers the third choice: NSD on top of IBM's VSD devices.

Network Shared Disk with dedicated NSD server(s)

In this model the application servers running the GPFS daemon are connected to the NSD servers, which provide access to the disk. Therefore, these NSD servers act as storage nodes. For availability reasons, normally two paired NSD servers with twin tailed disk access are used, providing access to the disks and balancing the workload between each other (optional, user configurable).

It is also possible to specify only one server for disk access (single NSD server). As the application servers need to get all data over the NSD servers, a high-speed, low latency network, like Gigabit Ethernet, Myrinet, or Infiniband, is required. This model is suited where the application servers act as compute nodes, and the application can separate tasks handled by each compute node independently. The compute nodes receive the data to fulfill the task from the central storage, compute the task, and send the result back to the central repository. In Digital Media environments, a rendering farm is a good example for this type of workload.

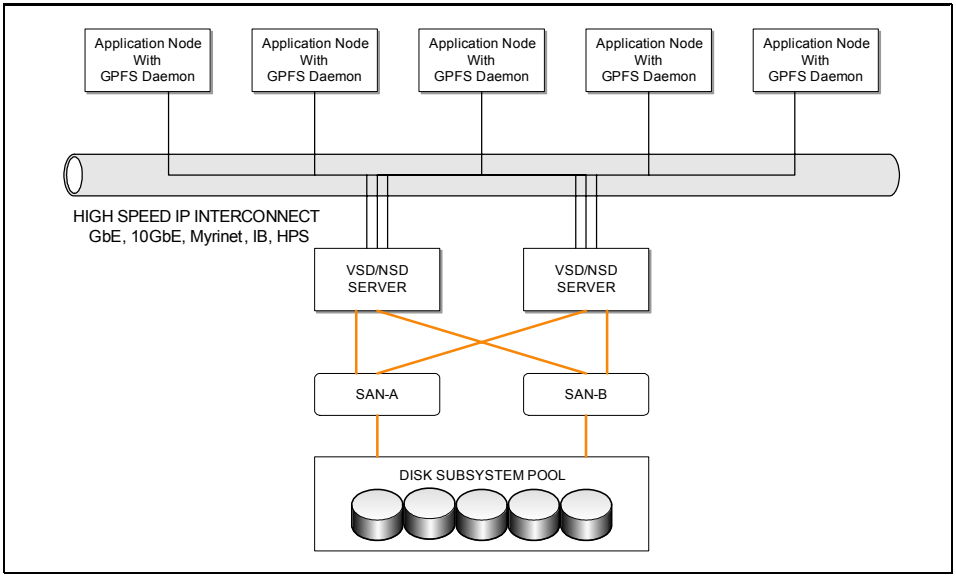

Figure 2-21 GPFS NSD/VSD model

Advantages:

This disk access model can be used in high performance computing environments where the compute nodes do not have locally attached storage (like the new IBM @server Blue Gene® Solution).

- Can utilize extra switch bandwidth.
- Performance scales with the number of servers.
- Uses un-used compute cycles if node utilization is less 100%.
- Works well with disks subsystem that can only attach to limited number of nodes.

Disadvantages:

As a single node is used for routing all data traffic from the client nodes to the storage, this may become a performance bottleneck.

- Performance can be limited by adapters in the servers (i.e., cannot make use of the full bandwidth of the disk subsystem).
- Recommended to use with dedicated storage servers (i.e., no other application should run on these servers).
- Compute cycles are stolen from the compute nodes - as they wait for I/O operations to complete (e.g., can seriously degrade performance for implicitly synchronous applications).

Direct attached model

Instead of associating a subset of disks with a particular server (pair), all disks are attached to all nodes within the cluster (currently via a SAN). Unlike other SAN based storage I/O products, GPFS does not use a centralized metadata server, as GPFS's metadata operations are generally distributed among cluster nodes, allowing it to scale up to larger SAN clusters. This model is used in broadcasting environments.

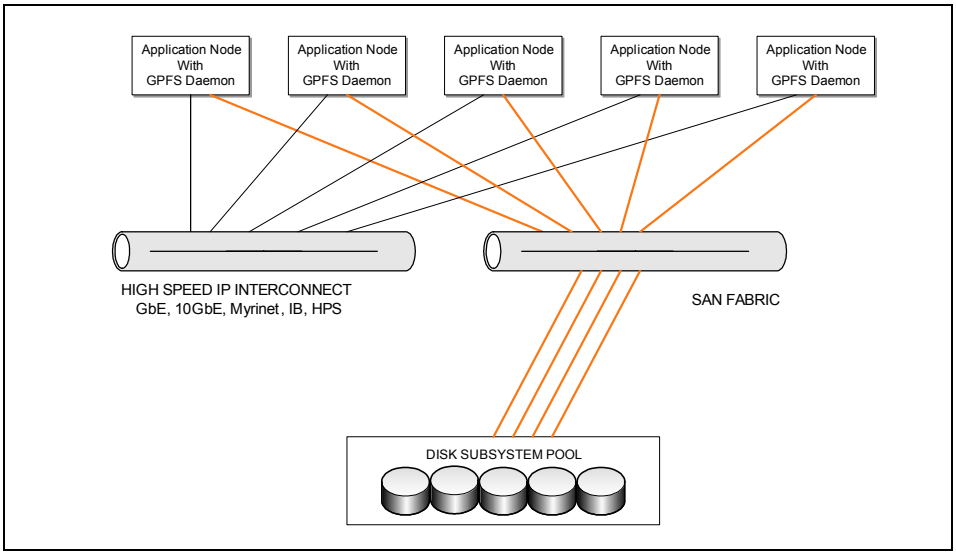

Figure 2-22 GPFS direct attached model

Advantages:

- FC operations are more efficient than IP operations (e.g., Myrinet).
- Simpler I/O model: a storage I/O operation does not require an associated network I/O operation (n.b., all LUNs are local to all nodes in a SAN).
- Can be used without a high-speed IP network switch (like, Myrinet, or SP Switch)

Disadvantages:

- Cost and complexity when building large SANs.

Mixed model (NSD and direct attached servers)

This model is a combination of the previously described models. As seen in Figure 2-23 on page 67, the three nodes have direct access to the disks, while the first two are specified as NSD primary and backup servers. The third node acts as application node and has also direct access to the disks (via SAN). A good example for an application is Tivoli Storage Manager (TSM). In this case, TSM can directly backup and restore all GPFS file systems of this cluster.

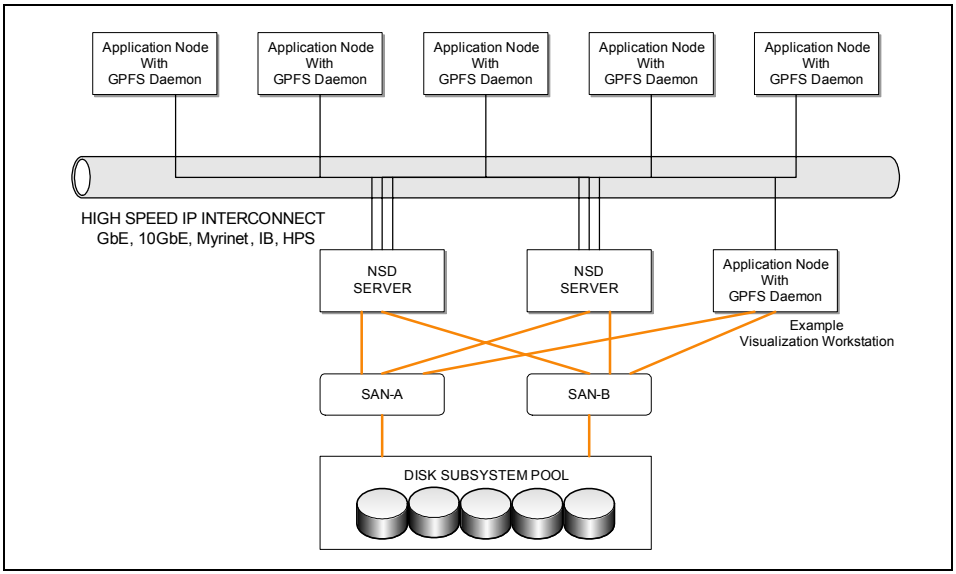

Figure 2-23 GPFS - mixed (NSD and direct attached) model

Joined model (heterogeneous NSD - AIX and Linux servers)

This model shown in Figure 2-24 on page 68 uses both platforms (AIX and Linux) as storage nodes (NSD servers). The main difference between this configuration and the previously mentioned models, is that in this case both AIX and Linux nodes are NSD servers for separate sets of disks and NSD clients to each other (still the same GPFS cluster). The storage nodes from the previous models were always homogenous (same operating system and architecture).

Attention: Due to the differences between AIX and Linux (i.e., disk device drivers), it is not possible to access directly the same disks from different operating systems. Therefore, it is mandatory to “isolate” the disks at SAN level (different SANs for AIX and Linux, or different zones inside the same SAN), and storage level (storage partitioning and LUN masking).

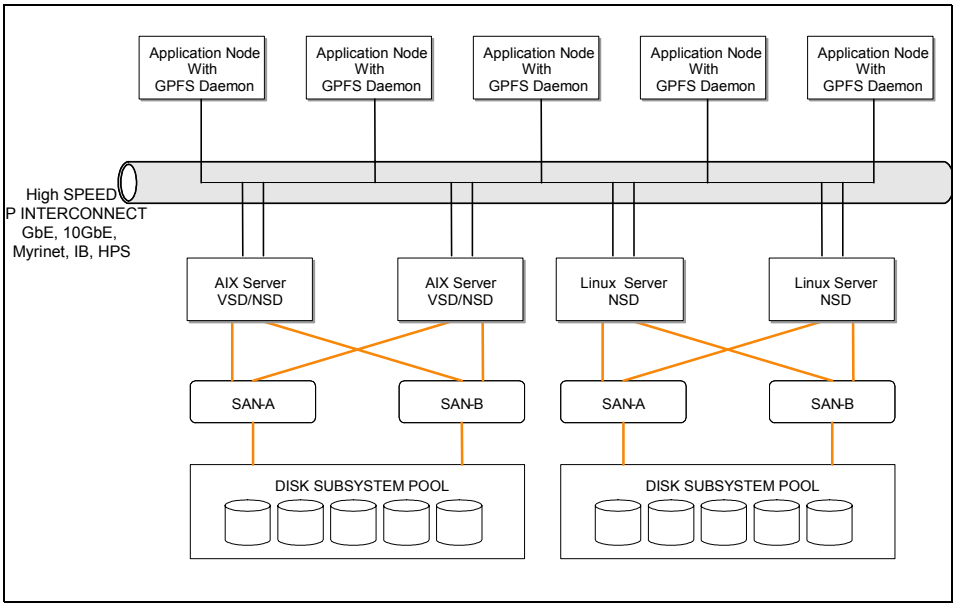

Figure 2-24 GPFS joined (AIX and Linux platform) model

Cross-cluster file systems access model

Finally, the GPFS cross-cluster mounted model allows multiple clusters to exchange data using GPFS functions (see Figure 2-25 on page 69). The advantage here is, that every cluster can be managed separately by different administrators, but users can still access file systems which are not defined in the local cluster. This is a replacement for the previous GPFS model with one cluster and multiple node groups (dropped in GPFS 2.3).

The clusters only need to initially exchange the cluster security keys and make the various file systems known to the other clusters. While each cluster configuration stays the same, there is no other interaction needed. Therefore, end users from all clusters can access all file systems belonging to other clusters as well, depending on the implementation and permissions configured.

Note: Since each cluster is managed independently, there is no automatic coordination and propagation of changes between clusters (like inside the same cluster). This means that if you have to be careful when deleting or changing nodes from a cluster, or you may affect access from remote GPFS clusters. You should notify the administrators of remote GPFS clusters to update the configuration.

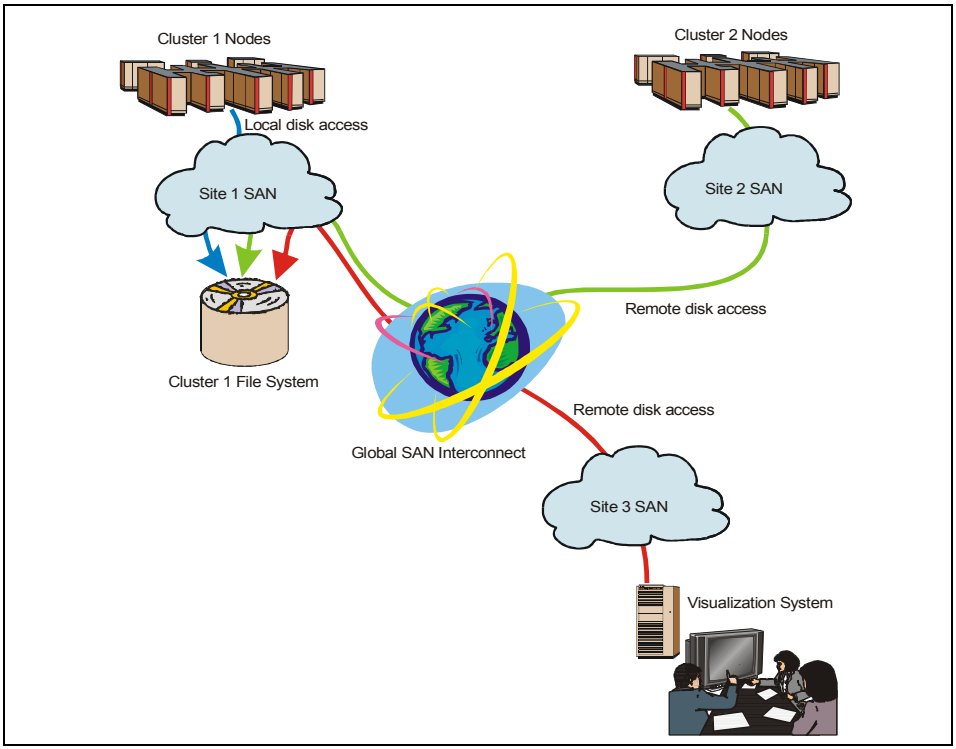

Figure 2-25 GPFS cross-cluster mount model

2.3 Designing your cluster

This section introduces the most important design points to consider when planning your GPFS cluster for a digital media environment. General design topics are also covered, as a good design will lead to a seamless implementation and a trouble-free cluster.

2.3.1 Knowing your environment — questions to ask

Like in any environment, acquiring the as much information as possible information prior to designing and implementing the solutions is a key point for the success of the project, and the best way to get this information is by asking questions. We can divide the questions that should be asked into four major topics:

- Infrastructure (capacity, performance, availability)
- Client attachment (operating system, network, protocols, block size, etc.)

- Application (compatibility, block size, dependencies)
- Miscellaneous (financial, legal, support contract)

In the following paragraphs we present a collection of questions covering the four topics. The list of questions we propose is not exhaustive, and it should be rather considered as a starting point to build up your own questions list. The more questions you ask, the better you can understand the requirements for the final solution.

Infrastructure questions

- What amount of storage capacity (file system) is needed?
- What is the expected overall performance (network / storage)?
- What capacity and performance growth will be expected?
- What are the requirements for availability and fault resilience?
- Is there a need for a disaster recovery solution?
- Which are the fail over timing requirements?
- What is the expected total throughput of the DMC system?
- Are there requirements for a “must have” throughput under all circumstances from the DMC system?
- Is there a need for an integrated Backup/Archive solution?
- and so on

Client attachment questions

- How many concurrent clients will access the system?
- How many (if any) direct attached clients will be also members of the DMC cluster? (this is only valid for AIX or Linux based machines)
- What is the expected throughput for one network-attached client?
- Are there any timing requirements from the applications side?
- What client access protocol (SAMBA, NFS, FTP, etc.) will be used?

Application related questions

- Which type of workload will be expected from the application side? (read/write ratio, random and/or sequential, many small files or very large files, parallel file access from the clients, etc.)
- What is the most commonly used block size from user applications?

Miscellaneous questions

- Is education needed for post implementation system operation?
- Are services needed (migration, implementation, maintenance)?
- Need for remote access / call home features / security?
- And maybe the most important questions: What are cost expectations (initial price, ongoing operation costs, etc.) for the entire solution?

Note: The question list is not exhaustive. Assessing the customer environment may require additional questions.

2.3.2 GPFS considerations

For reference and other configuration specific issues, see also *General Parallel File System (GPFS) for Clusters: Concepts, Planning, and Installation*, GA22-7968.

For performance reasons GPFS is striping all files over all disks of a file system. This means that from the storage (disk) perspective the access pattern is random. This is also independent of which GPFS striping method was chosen. Therefore, we recommend (at least for DM environments, but not only) the following:

- Use storage RAID5 with 4+P or 8+P for data
- Is recommended to create one LUN per storage array
- For data arrays: in DS4xxx storage subsystems you should disable read ahead mechanism and read/write caching.
- Select a segment/stripe size which matches the GPFS block size
- Assign separate physical disks (arrays) for metadata
- For metadata arrays only: in DS4xxx storage subsystems, enable read ahead mechanism and write cache mirroring
- To match performance, select correct disk (type, size, rotation speed)

2.3.3 Storage design considerations

- Choose the appropriate disk storage subsystem
  - As en example: DS4300 provides good price/performance/capacity mix
  - DS4300Turbo: in case you need to extend the capacity of the base configuration (other improved capabilities of the Turbo version are not really used in a Digital Media environment).

- DS4500: At the time this book was written, this was considered as the “work horse” of the IBM storage subsystems for DM environments, with good expandability and configuration flexibility.
- DS6000: was not yet tested for DM environments at the time of this writing.
- ESS800/DS8000: Although the performance of these storage subsystems is much better that DS4xxx series, this type of storage needs to be carefully considered, as it is targeted for large enterprise environments, and the cost of such systems may not make them suitable for DM.

Note: Even though the disk subsystem may be officially supported for GPFS in general, in Digital Media, the solution has to be VALIDATED for each specific environment.

- Use different Host Bus Adapters (HBA) for disk and tape
- Is Hierarchical Storage Management (HSM) or other form of disk space management needed?

2.3.4 Network and miscellaneous design considerations

- Keep in mind the performance impact the array rebuild process may introduce in case a disk fails and is replaced.
- Make sure to use non-blocking network components (switches)
- It is recommended to use a separate (dedicated) network for cross-cluster mounted file systems (between two separate clusters).
- If possible, for NSD/VSD implementation, use a high speed, low latency network like IBM HPS, Myrinet, or Infiniband
- Choose the suited platform and operating system (application and price dependent).

2.3.5 Classification of GPFS commands

This section is an overview of the GPFS management activities and the tools needed to perform these tasks. This section will provide information about these topics:

- Classifying GPFS commands (cluster, disk, node, performance-related).
- Table of GPFS commands influencing/impacting performance.
- Table of GPFS commands, which are disruptive when used.

Table 2-2 provides a quick overview about the function of any GPFS command and its influence in the operation and answers the following questions:

- Is the command influencing the whole cluster or the individual node configuration only?
- Which configuration (cluster, node, file system, disk, or file) does the command change?
- Is the command affecting performance of the system while active?
- Is the command disruptive in the sense, that a component or the whole cluster is affected and needs to be offline.
- Is the command a script or a binary file?

To better understand how we classification of the commands, we describe the usage of the table for **`mmadddisk`** command, the first command in Table 2-2.

`mmadddisk` command:
This command adds previously defined NSD disks to an existing file system in a GPFS cluster, and can be issued from any node within the cluster. This command affects the GPFS environment on *all* nodes in a cluster. Also, this command modifies a file system, therefore it is *file system related*. It may also affect the file system *performance* when used with the "-r" option (see Table 2-3 on page 76). That means the command has direct impact (temporary) on the performance of the system *while the command is running*.

It does not mean that this command changes the overall performance behavior (overall throughput in the sense of tuning) of the GPFS cluster or the designated file system.

Also, this command *is not disruptive*, as the cluster and the affected file system is operable at all time while running this command.

This command is in fact a *shell script* (as reported by the unix **`file`** command.

Table 2-2 Classification of GPFS commands

command	node influencing / node related	cluster related	file system related	disk related	file related	affecting FS performance[a]	disruptive yes/no[b]	script / binary
`mmadddisk`			X	X		yes	no	S
`mmaddnode`		X				no	no	S
`mmauth`		X	X			no	yes	S

command	node influencing / node related	cluster related	file system related	disk related	file related	affecting FS performance[a]	disruptive yes/no[b]	script / binary
mmbackup	X		X			yes	no	S
mmchattr					X	no	no	B
mmchcluster		X				no	no	S
mmchconfig	X	X				no	yes	S
mmchdisk				X		yes	no	S
mmcheckquota			X		X	yes	no	B
mmchfs			X			no	yes	S
mmchmgr	X		X			no	no	S
mmchnsd				X		no	yes	S
mmcrcluster		X				no	no	S
mmcrfs			X			no	no	S
mmcrnsd				X		no	no	S
mmcrsnapshot			X			yes	no	S
mmcrvsd				X		no	no	AIX
mmdefedquota			X		X	no	no	B
mmdefquotaoff			X		X	no	no	B
mmdefquotaon			X		X	no	no	B
mmdefragfs			X			yes	no	S
mmdelacl					X	no	no	B
mmdeldisk			X	X		yes	no	S
mmdelfs			X			no	no	S
mmdelnode	X	X				no	no	S
mmdelnsd				X		no	no	S
mmdelsnapshot			X			no	no	S

command	node influencing / node related	cluster related	file system related	disk related	file related	affecting FS performance[a]	disruptive yes/no[b]	script / binary
mmdf			X			yes	no	S
mmeditacl					X	no	no	B
mmedquota			X			no	no	B
mmexportfs			X			no	yes	S
mmfsck			X			yes	yes	S
mmfsctl			X			yes	yes	S
mmgetacl					X	no	no	B
mmgetstate	X	X				no	no	S
mmimportfs			X			no	no	S
mmlsattr					X	no	no	B
mmlscluster		X				no	no	S
mmlsconfig		X				no	no	S
mmlsdisk			X	X		no	no	S
mmlsfs			X			no	no	S
mmlsmgr			X			no	no	S
mmlsnsd			X	X		no	no	S
mmlsquota			X			no	no	S
mmlssnapshot			X			no	no	S
mmpmon	X		X			yes	no	S
mmputacl					X	no	no	B
mmquotaoff			X		X	no	no	B
mmquotaon			X		X	no	no	B
mmremotecluster		X				no	no	S
mmremotefs			X			no	no	S

command	node influencing / node related	cluster related	file system related	disk related	file related	affecting FS performance[a]	disruptive yes/no[b]	script / binary
mmrepquota			X			no	no	B
mmrestorefs			X			no	yes	S
mmrestripefs			X			yes	no	S
mmrpldisk			X	X		yes	no	S
mmsanrepairfs			X			no	no	B
mmshutdown	X	X				no	yes	S
mmsnapdir			X			no	no	S
mmstartup	X	X				no	no	S

a. See Table 2-3 on page 76 for further details
b. See Table 2-4 on page 78 for further details

In addition to Table 2-2 on page 73, which only shows that some commands are impacting the performance (while the system is running), Table 2-3 shows the performance affecting commands when using specific parameters. Keep in mind that the commands may be issued without any performance impact while not used in conjunction with the parameter listed in the table.

Table 2-3 Explanation of performance impacting command parameters

performance impacting commands	-/ --	parameter	explanation
mmadddisk	--	-r	re striping of file system decreases performance while nodes and storage are used to fulfill this operation.
mmbackup	-	all	by default.
mmchdisk	--	stop, start, resume, suspend	disk maintenance is by default a performance impacting operation. Re striping of file system is needed after a disk failed or stopped.
mmcheckquota	--	all	The mmcheckquota command is I/O intensive and should be run when the system load is light.

performance impacting commands	-/ --	parameter	explanation
mmcrsnapshot	-		A snapshot is a copy of the changed user data in the file system. System data and existing snapshots are not copied.
mmdefragfs	-		The mmdefragfs command moves existing file system data within a disk to make more efficient use of disk blocks. The
mmdeldisk	--	-r	Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.
mmdf	--		The command is I/O intensive and should be run when the system load is light.
mmfsctl	-	suspend	Instructs GPFS to flush the internal buffers on all nodes, bring the file system to a consistent state on disk, and suspend the processing of all subsequent application I/O requests.
mmpmon	-		Use the mmpmon command to manage GPFS performance monitoring functions and display performance monitoring data.The performance monitoring request is sent to the GPFS daemon running on the same node that is running the mmpmon command.
mmrestripefs	--		Rebalancing of files is an I/O intensive and time consuming operation, and is important only for file systems with large files that are mostly invariant. In many cases, normal file update and creation will rebalance your file system over time, without the cost of the rebalancing.

performance impacting commands	-/ --	parameter	explanation
mmrpldisk	--		All data on the old disk is migrated to the new one.

As Table 2-2 on page 73 only shows that some commands may be disruptive, Table 2-4 lists the commands and the parameters that cause disruptions. Further an explanation is provided for which component(s) is/are affected, as not always the whole cluster needs to force unmount a file system or to shutdown.

Table 2-4 Explanation of disruptive commands

disruptive commands	parameter	explanation
mmauth	genkey	Generates a new public/private key pair for this cluster. For this option to succeed, GPFS must be inactive on all nodes in the cluster.
mmchconfig (node only)	designation	GPFS must be stopped on any quorum node that is being changed from quorum to non-quorum, but not vice versa.
mmchconfig (all nodes in the cluster)	tiebreaker disk	When changing the “tiebreaker Disks”, GPFS must be down on all nodes in the cluster.
mmchdisk (disk or if not protected whole file system)	start, stop, suspend, resume	Stopping/starting a disk has a direct influence in the whole system behavior. After stopping a disk the file system will be re striped after the disk comes back online.
mmchfs (file system only)	-T mount point	The file system must be unmounted on all nodes prior to issuing the command.
	-Q {yes | no}	To activate/deactivate quota management the file system needs to be unmounted and remounted.
mmchnsd (file system only)		The file system that contains the NSD being changed must be unmounted prior to issuing the mmchnsd command.

disruptive commands	parameter	explanation
mmfsck		The file system must be unmounted before you can run the mmfsck command with any option other than -o.
mmrestorefs		You must unmount the file system from all nodes in the cluster. The file system may not be remounted until the mmrestorefs command has successfully completed

General hints / advice for GPFS administration

For detailed advice and more hints, check also *General Parallel File System (GPFS) for Clusters: Administration and Programming Reference*, SA22-7967.

- You must designate at least one node as a quorum node.
- We strongly recommend to designate the cluster configuration data servers as quorum nodes.
- Always try to run the **mmdeldisk** command when system load is low.
- Exercise caution when deleting quorum nodes from the GPFS cluster. If the number of remaining quorum nodes falls below the requirement for a quorum, you will be unable to perform file system operations.
- The command **mmdf** is I/O intensive and should be run when the system load is low.
- When setting quota limits for a file system, replication within the file system should be considered.

GPFS limitations

- There is a maximum limit of 31 snapshots per file system. Snapshots may be deleted only by issuing the **mmdelsnapshot** command. The `.snapshots` directory cannot be deleted.
- There is a maximum of 32 file systems per cluster

Refer to GPFS documentation for more detailed descriptions.

3

Infrastructure configuration

This chapter is an example about how to set up the basic hardware and software for implementing a GPFS cluster, following a nine-step approach. The four main areas to configure are:

- Hardware (server, storage)
- Networking (LAN/SAN switches)
- Software (operating system, drivers)
- GPFS installation and configuration

3.1 General considerations for installing GPFS

This section describes the steps necessary to achieve a GPFS cluster. The preliminary configuration for a GPFS cluster includes hardware and software. By correctly preparing the environment for GPFS, the GPFS software installation and configuration will be very easy and can be performed in just 20 minutes.

The following list contains the nine steps needed for a successful GPFS implementation:

1. Nodes properly installed (hardware, software, device drivers)
2. Proper network configuration (IP, switches)
3. Node-to-node remote command execution and time synchronization.
4. Disks to servers/nodes attachment configuration and verification

Note: Steps 1 through 4 are general steps for any cluster implementation. Getting the proper hardware and software running before creating your cluster the key for a seamless GPFS configuration.

5. Install GPFS packages and define a GPFS cluster
6. Start GPFS and verify all nodes join
7. Create VSDs/NSDs
8. Create file systems
9. Mount file systems

Steps 1 to 4 (GPFS prerequisites) will be described in these sections:

- 3.2, “Hardware setup and configuration” on page 83
- 3.3, “Software installation (Linux-based cluster)” on page 103

And the GPFS installation and configuration (steps 5 to 9) will be described in:

- 3.4, “GPFS installation and configuration” on page 126

Before starting, here are some general recommendations:

Tip: Use meaningful and consistent naming conventions for network, nodes, disks, file systems.

- It is much easier to configure and administer a cluster with a consistent and self explaining naming space throughout the environment.

Tip: Assign a management station (one of the GPFS cluster nodes). This helps in increasing productivity and consistency for administrative operations.

- Even though is possible to administer a GPFS cluster from any node in the cluster, it is generally a good idea to assign one of the nodes (one of the quorum nodes) as a management workstation for your GPFS cluster.
- As you work with your cluster configuration and maintenance, you may create your own environment variables, automation scripts and/or descriptor files. Since there is no automated method to keep these (user created) files synchronized in sync (e.g., a file collection mechanism) on all the nodes, performing the operations from another node may not come at hand at all times.
- Keep in mind to have another station also prepared in case your primary management station fails.

3.2 Hardware setup and configuration

The foundation for a well designed and configured GPFS cluster is a properly chosen and configured hardware platform. In this section we present the hardware platform we have used for testing in our environment (ITSO lab).

3.2.1 List of components

Table 3-1 shows every component by its role, number of devices available, the device name, and finally the microcode and other software versions for each of the elements we used during our tests.

Table 3-1 List of components used in the ITSO lab

System component	#	Type	Feature
Storage Controller	1	DS4500	Firmware version: 06.12.03.00 NVSRAM version: N1742F900R912V06
Disk Expansion	2	EXP700	Firmware 9327 28x drives 72 GB 15k rpm B356/954
FC-switch	2	2109-F32	Fabric operating system V4.1.2

System component	#	Type	Feature
Hosts	8	p630	Server nodes GPFS cluster: GPFS-ITSO3, GPFS-ITSO4 Operating system: AIX 5.3 ML1
	2	p550	Server nodes (Lpared) ⇒4 nodes GPFS cluster: GPFS-ITSO1 Operating system: SLES9 kernel 2.6.5-7.139/145
	4	blade JS20	Server nodes GPFS cluster: NSD clients Operating system: SLES9 kernel 2.6.5-7.139/145
	2	p630	Server nodes GPFS cluster: GPFS-ITSO2 Operating system: SLES9 kernel 2.6.5-7.139/145
File system		GPFS	V2.3.0.3
File sharing		SAMBA	V3.0.1
LAN switch			Gigabit Ethernet Switches

For an overview and for explanation purposes, we use a subset of components to describe the setup. It is easier to explain, in the following sections, how the configuration has to be performed. The subset of components we will use contains:

- Two SAN switches 2109-F32
- DS4500 (including expansion units) with four host-side minihubs and two connections to each switch
- Two IBM @server pSeries 550 servers, while using the LPAR feature (for a total of four GPFS nodes)

3.2.2 Disk storage devices

IBM DS4500 (formerly FAStT900) delivers break-through disk performance and outstanding reliability for demanding applications in data-intensive computing environments. The DS4500 is an effective disk system for any enterprise seeking performance without borders. In addition, we used the IBM TotalStorage DS4000 EXP700 Fibre Channel Storage Expansion Unit.

In new actual implementations the EXP710 should be used instead, if possible, as it offers several enhancements, including improved reliability and efficiency utilizing internal switch technology to attach to each disk drive.

The complete list of IBM storage devices and subsystem can be found at:

http://www.storage.ibm.com

We have configured the storage prior to configuring GPFS (nodes, SAN switches, storage):

- Cabling the environment (storage connections for SAN and LAN)
- Check for the latest microcode and management software updates
- Assign a system as the management station and installed Storage Manager software.
- Configure the box for remote management (IP configuration)
- Use the Storage Manager Client (GUI) or the Storage Manager Command Line Interface (SMcli) to configure the storage
  - Licensing (add the license keys)
  - LUN configuration
  - Storage partitioning (LUN masking)

Before using the device, we checked for latest software updates, which can be found at:

http://www-1.ibm.com/servers/storage/support/disk/ds4500/firmware1.html

Attention: Make sure to use the correct files for your platform.

Figure 3-1 on page 86 shows the cabling between the FC-switches, storage controller (IBM TotalStorage® DS4500) and the expansion units used in our lab environment.

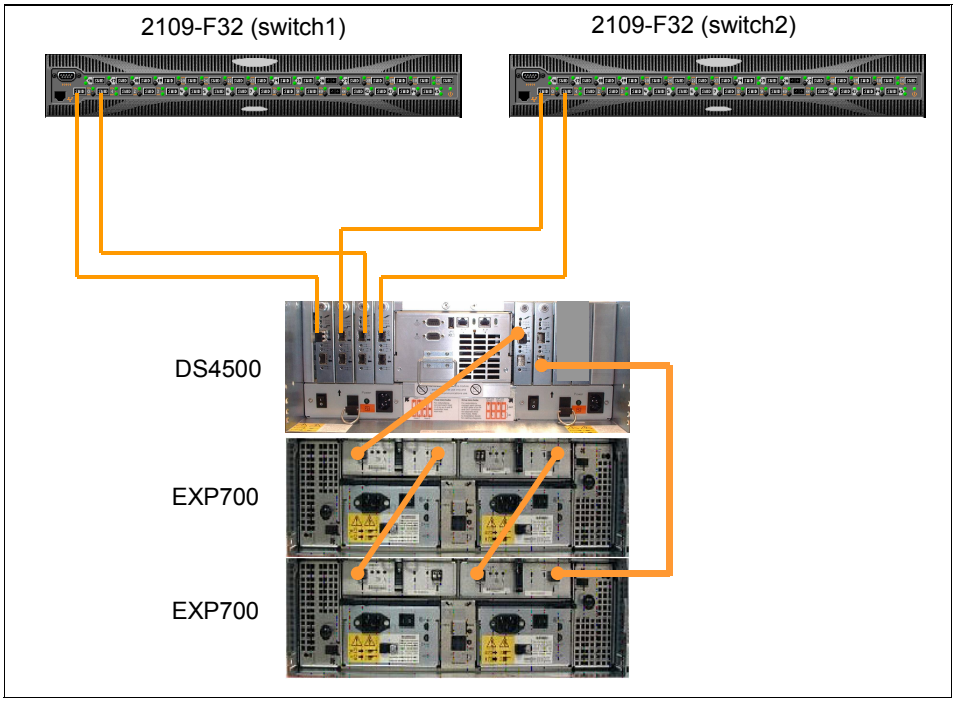

Figure 3-1 SAN Cabling of switches, storage controller and expansion units

After updating the microcode and cabling FC connections to the box, the DS4500 needs to be configured. Example 3-2 on page 89 shows the LUN configuration we used in our IBM TotalStorage® DS4500.

From the management workstation (where the Storage Manager software has been installed) you have two choices to create and store storage subsystem profile. You can either use the SM GUI or the SMcli (command line interface). For simplicity, we start configuring the storage using the GUI (see Figure 3-2 on page 87).

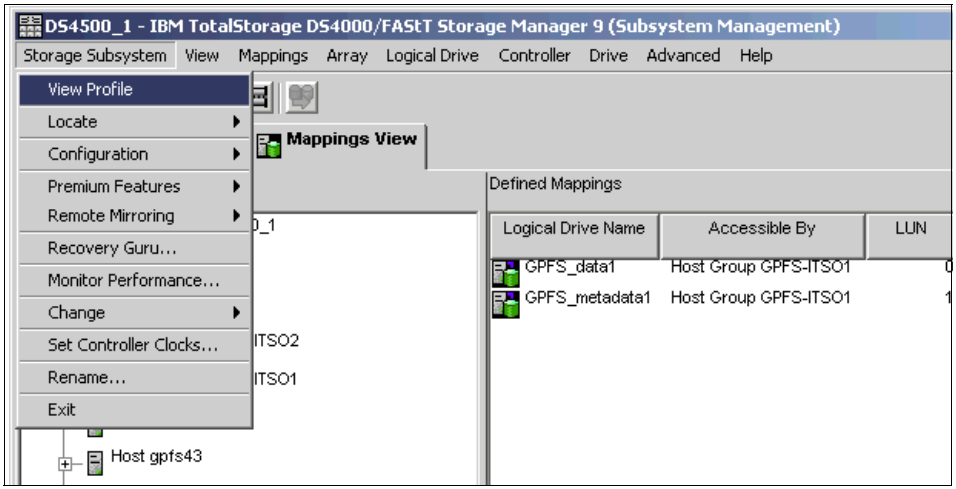

Figure 3-2 DS4500 GUI - Check the Subsystem Storage Profile

Figure 3-3 shows the entry screen of the GUI based profile. Here you find all information organized by tabs for a better overview.

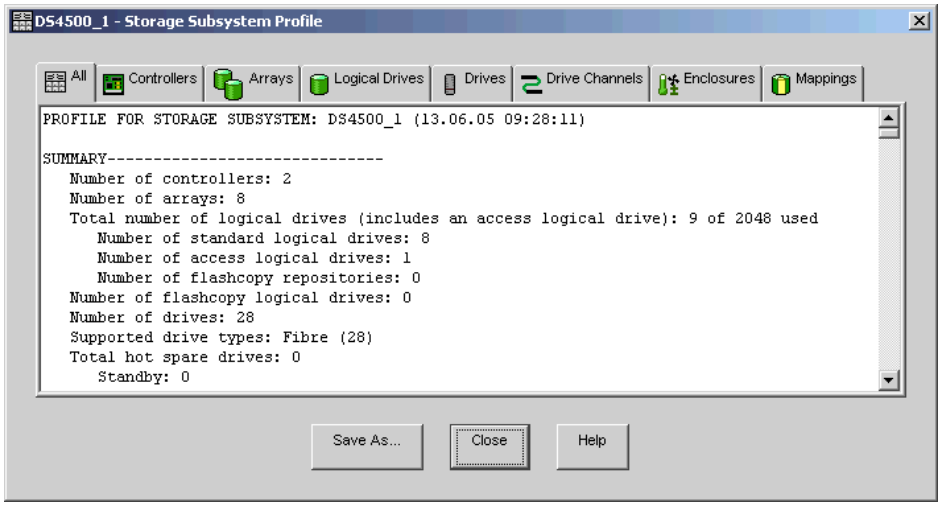

Figure 3-3 DS4500 GUI - Storage Subsystem Profile window

You can also save the profile in text format, by pressing the “Save as” button, and entering the destination file to be saved (see Figure 3-4 on page 88).

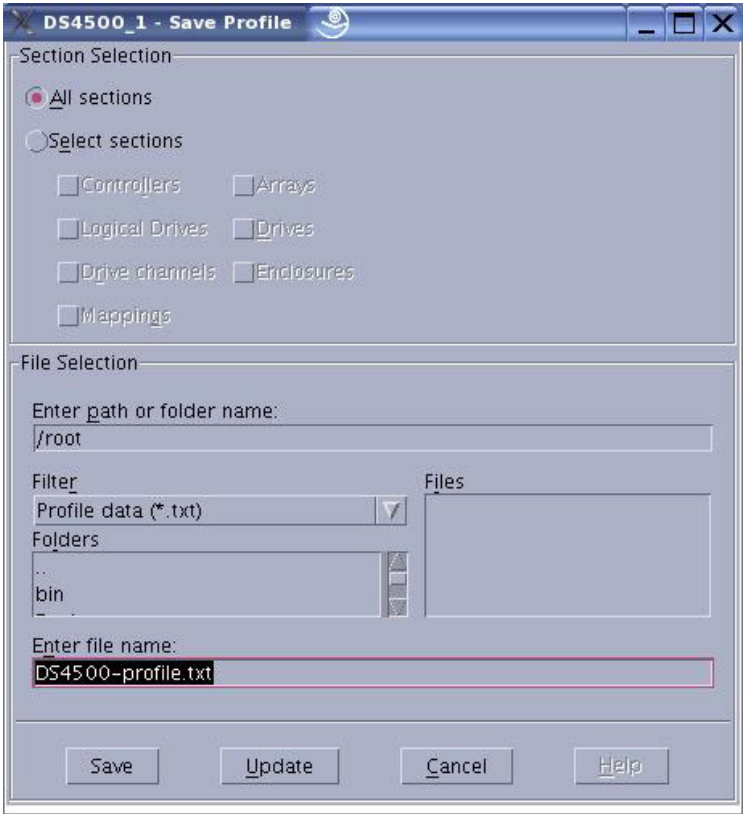

Figure 3-4 DS4500 - Save the storage subsystem profile

The second option is the Storage Manager command line interface (SMcli). The command needed is shown in Example 3-1. For this example we have used a Linux workstation, therefore the **SMcli** executable is located in **/usr/bin/SMcli**. On AIX you will find the command in **/usr/SMclient/SMcli**. You need to specify the IP-address or the hostname of your DS4500 controller. We have redirected the output to a file (`/config/DS4500.txt`).

Example 3-1 Using SMcli to get the storage subsystem profile

```
# Linux path: /usr/bin/SMcli
SMcli 192.168.100.21 -c "show storagesubsystem profile;" >/config/DS4500.txt
```

Example 3-2 on page 89 shows the relevant components of the storage profile for the DS4500 we used in our lab. The most important settings will be explained in detail in the following paragraphs.

Example 3-2 DS4500 storage profile - partly (showing the LUN configuration)

```
PROFILE FOR STORAGE SUBSYSTEM: DS4500_1 (09.06.05 11:34:32)

SUMMARY------------------------------
   Number of controllers: 2
   Number of arrays: 8
...........>>> Omitted lines <<<.............
CONTROLLERS--------------------------------
   Number of controllers: 2

...........>>> Omitted lines <<<.............
ARRAYS-------------------------------
   Number of arrays: 8
...........>>> Omitted lines <<<.............

STANDARD LOGICAL DRIVES-----------------------------

SUMMARY
   Number of standard logical drives: 8
   See other Logical Drives sub-tabs for premium feature information.

   NAME            STATUS   CAPACITY    RAID LEVEL  ARRAY
   GPFS_data1      Optimal  271,463 GB  5           3
   GPFS_data2      Optimal  271,463 GB  5           4
   GPFS_data3      Optimal  271,463 GB  5           1
   GPFS_data4      Optimal  271,463 GB  5           2
   GPFS_metadata1  Optimal  67,866 GB   1           5
   GPFS_metadata2  Optimal  67,866 GB   1           6
   GPFS_metadata3  Optimal  67,866 GB   1           7
   GPFS_metadata4  Optimal  67,866 GB   1           8

DETAILS
   Logical Drive name: GPFS_data1
...........>>> Omitted lines <<<.............
...
   Logical Drive name: GPFS_metadata1
...........>>> Omitted lines <<<.............
...
```

As described already in 2.2.3, “GPFS block allocation” on page 38, the GPFS file system block size (1 MB planned) should be matched with the segment size used by the storage system. To obtain the best performance we configured the IBM TotalStorage® DS4500 as follows:

- RAID5 arrays with five disks using a 4+P scheme.
- Cache block size: 16 KB

- Configuration for **data** LUNs:
  - Segment size: 256 KB (to match the GPFS block size of *4 x 256kB = 1MB*)
  - Read cache disabled
  - Write cache, write cache mirroring, and write cache without batteries are all disabled
  - Modification priority: low
- Configuration for **metadata** LUNs:
  - Segment size: 64 KB
  - Read cache enabled
  - Read ahead multiplier: 5
  - Write cache, write cache with mirroring, and write cache without batteries are all enabled. Make sure the batteries are working properly.
  - Modification priority: low

Attention: If the data availability requirement is not critical, write performance can be improved significantly by enabling write caching. The drawback is that the DS4500 maintains a volatile disk cache distributed between two distinct RAID controllers. When a write operation is completed (including flushing buffers), control is returned to the application when the data is written to this cache, but not yet committed to the disk.

If one of these RAID controllers fails, then the data in the associated cache will be lost; since this can include metadata, the entire file system can be corrupted. Given the potentially large size of GPFS file systems (for example, 100s of terabytes or more) and the fact that files are striped over all of the disks, the magnitude of the risk is multiplied. Enabling write cache mirroring can compensate for this risk, but may compromise storage performance associated with write caching, because the cache synchronization takes place on the backend disk loops (there is no dedicated bus for this operation).

For high availability, the disks of a particular LUN should be distributed over as many disk enclosures as possible. Ideally, every enclosure should contain only one (1) disk of each RAID array. For example, for 4+P arrays you should use at least five expansion units and configure the array by using one drive in each expansion unit. This will protect from outages in case a complete expansion unit fails. In addition, the expansion units are cabled in such a way that two independent loops will have access to the same drives (see Figure 3-5 on page 91 for 4+P array configuration and Figure 3-6 on page 92 for back end cabling).

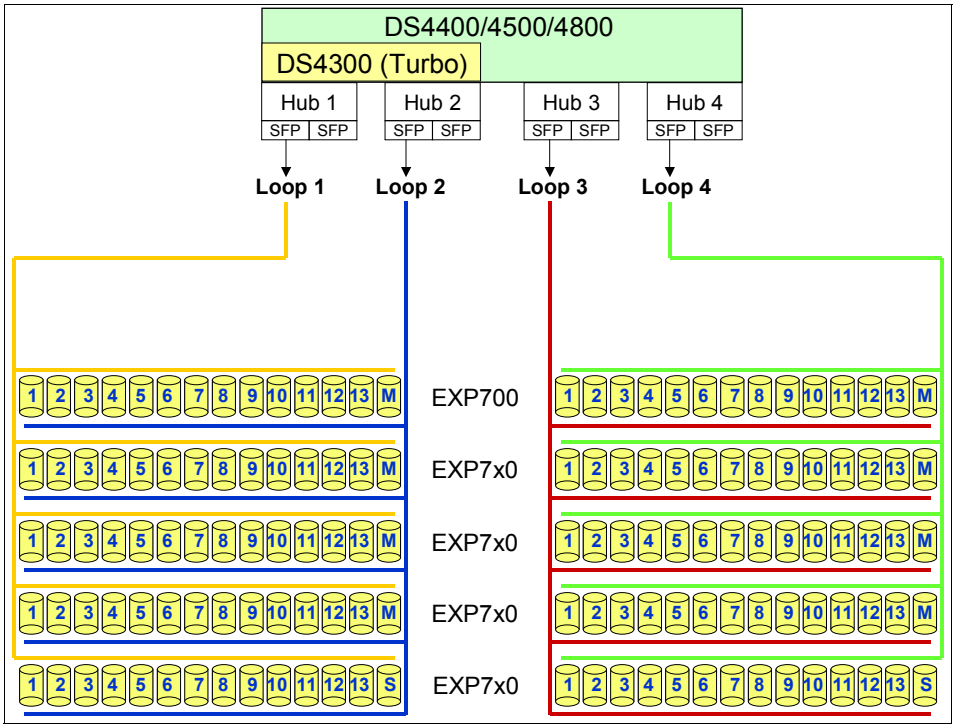

Figure 3-5 DS4x00 configuration example using 4+P arrays

For performance reasons, you should consider the following:

- Both controllers can access all backend loops

  This could lead to loop contention. To avoid that, all disks of a particular LUN need to be assigned/dedicated to the same controller.

- The disks in an enclosure must be connected to two different loops, e.g., in an enclosure disks are divided the following way:

  – disk1, disk3, disk5... to loop A

  – disk2, disk4, disk6... to loop B

  This is especially important if you do not have enough storage enclosures to distribute your disks across separate enclosures (you have to assign more than one disk within the same enclosure to the same array/LUN).

Note: For performance and availability reasons we suggest the following rules:

1. Create one LUN per array.
2. Choose disks out of the same loop to create an array.
3. Disks of a particular LUN should be distributed over as many disk enclosures as possible.
4. Distribute the LUNs equally over both controllers.

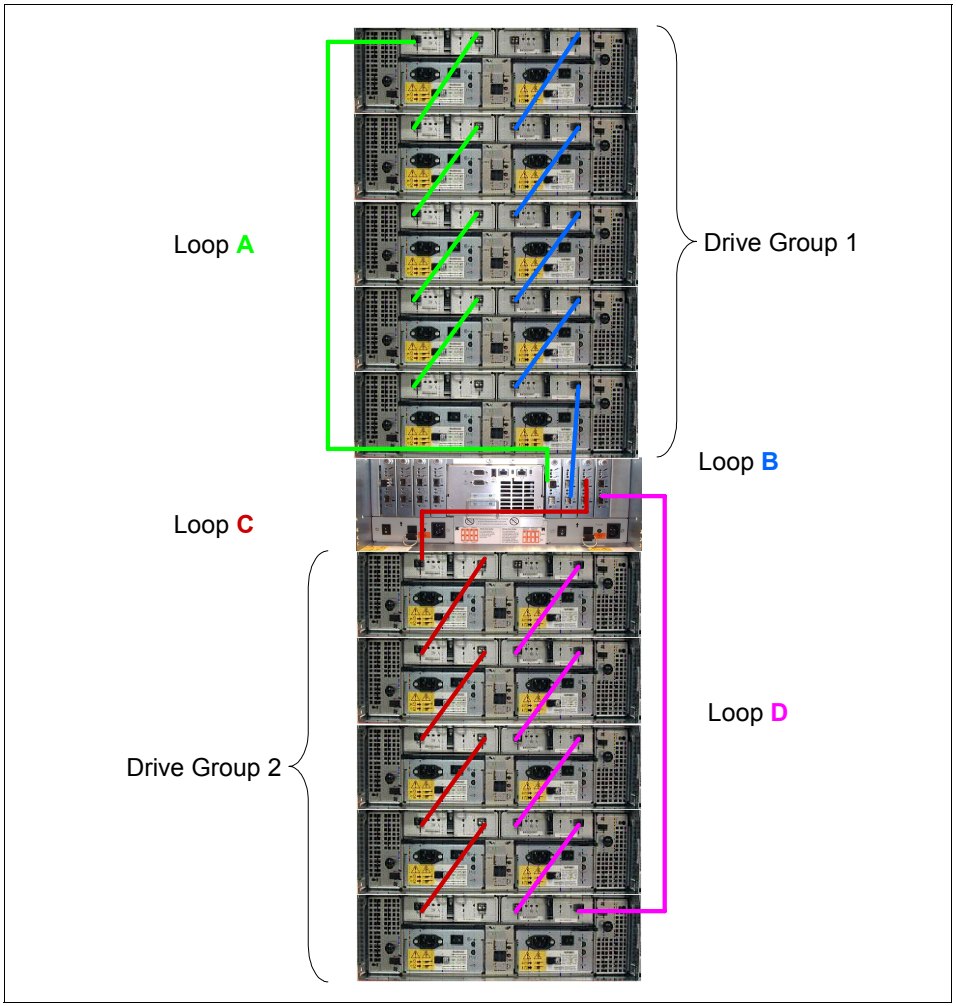

Figure 3-6 Optimal cabling example for DS4400/4500 with a total of 10 EXPs

Figure 3-7 shows the overall configuration of the storage controller, loops, expansions units, and LUN distribution for an 8+P configuration.

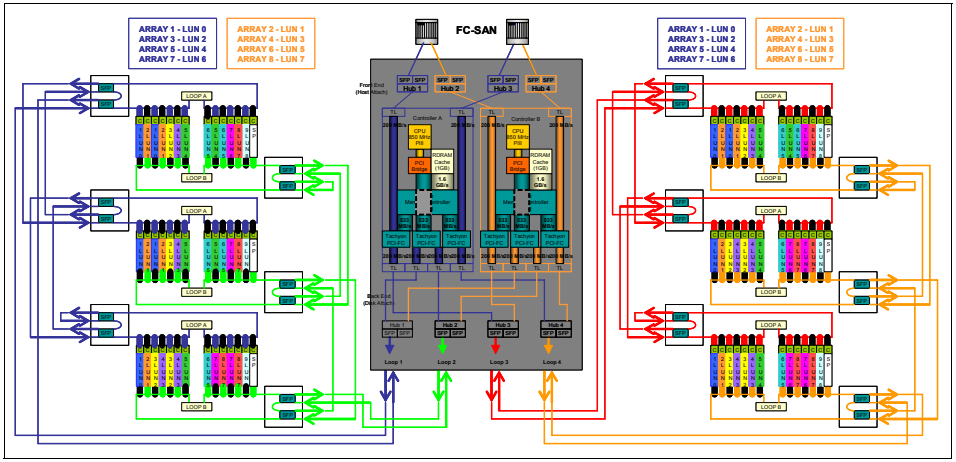

Figure 3-7 DS4500 configuration example for 8+P RAID5 arrays/LUNs

Note: Storage partitioning/mapping is needed to avoid data corruption specially when different operating systems are accessing the same storage.

Finally, we partitioned the storage. This is to protect LUNs for being accessed by other systems connected to the same SAN. In addition, zoning must be configured at the SAN level (see “Storage Area Network (SAN)” on page 96).

Example 3-3 shows the mappings for two host groups (GPFS-ITSO1 and GPFS-ITSO2). We partitioned the four available LUNs (data1, data2, metadata1 and metadata2), so that each host group can see one data and one metadata LUN. Host group GPFS-ITSO1 consists of the hosts gpfs41, gpfs42, gpfs43, and gpfs44.

All hosts of this host group are single attached only, which means having only one HBA available. Host group GPFS-ITSO2 consists only of two hosts gpfs29 and gpfs 30, but each host has two HBAs implemented.

Example 3-3 DS4500 storage profile - partly (showing the mappings of LUN-Host)

```
MAPPINGS (Storage Partitioning - Enabled (2 of 64 used))-------------------

   VOLUME NAME      LUN  CONTROLLER  ACCESSIBLE BY          VOLUME STATUS
   GPFS_data1       0    A           Host Group GPFS-ITSO1  Optimal
   GPFS_metadata1   1    A           Host Group GPFS-ITSO1  Optimal
   GPFS_data2       0    B           Host Group GPFS-ITSO2  Optimal
```

```
   GPFS_metadata2  1    B           Host Group GPFS-ITSO2  Optimal
...

   TOPOLOGY DEFINITIONS
      DEFAULT GROUP
         Default type: LNXCL
...
      HOST GROUP GPFS-ITSO1
         Host: gpfs41
            Host Port: 10:00:00:00:c9:42:1e:83
               Alias: gpfs41_1
               Type: LNXCL
            Host Port: 10:00:00:00:c9:42:26:9a
               Alias: gpfs41_2
               Type: LNXCL
         Host: gpfs42
            Host Port: 10:00:00:00:c9:42:2f:78
               Alias: gpfs42_1
               Type: LNXCL
            Host Port: 10:00:00:00:c9:42:2A:7a
               Alias: gpfs42_2
               Type: LNXCL
         Host: gpfs43
            Host Port: 10:00:00:00:c9:40:77:58
               Alias: gpfs43_1
               Type: LNXCL
            Host Port: 10:00:00:00:c9:40:77:8b
               Alias: gpfs43_2
               Type: LNXCL
         Host: gpfs44
            Host Port: 10:00:00:00:c9:40:3b:41
               Alias: gpfs44_1
               Type: LNXCL
            Host Port: 10:00:00:00:c9:40:1d:c7
               Alias: gpfs44_2
               Type: LNXCL

      HOST GROUP GPFS-ITSO2
         Host: gpfs29
            Host Port: 10:00:00:00:c9:31:b4:22
               Alias: gpfs29_1
               Type: Linux
            Host Port: 10:00:00:00:c9:31:b0:4c
               Alias: gpfs29_2
               Type: LNXCL
         Host: gpfs30
            Host Port: 10:00:00:00:c9:33:01:a1
               Alias: gpfs30_1
               Type: LNXCL
```

```
            Host Port: 10:00:00:00:c9:33:01:8a
               Alias: gpfs30_2
               Type: Linux
...
```

Attention: If you are using Linux RDAC/MPP (redundant disk array controller/multi path proxy) storage drivers on your nodes, you have to configure the DS4500 LUNs for “LNXCL” access type, which disables AVT/ADT (auto volume/drive transfer) for these LUNs.

By using the SM GUI to implement the mappings, five steps are needed:

1. **Create logical drive(s)**
   We create several LUNs to be used for GPFS as data and metadata LUNs. Therefore we chose the naming convention *data#* and *metadata#* as LUN names, where “#” is an integer value starting with 1.
2. **Define host groups**
   For ease of configuration we assign the same names used for the GPFS clusters, in our case GPFS-ITSO1 and GPFS-ITSO2
3. **Define the hosts under the host groups**
   We have used the host names assigned to the nodes:
   Cluster GPFS-ITSO1 contains the hosts gpfs41,gpfs42, gpfs43 and gpfs44.
   Cluster GPFS-ITSO2 contains the hosts gpfs29 and gpfs gpfs30.
4. **Define the host ports**
   We need to specify which HBA belongs to which host. The HBAs are identified by their World Wide Node Number (WWNN), and we created an alias for the each identifier (WWNN). We name the aliases by choosing the host name plus an increasing integer (0, 1, 2...) for multiple HBAs in the same node.
5. **Assign access for each host group or host**.
   This ia an access control lists which specifies which HBA is allowed to access which LUN. In our environment:
   Select Host Group GPFS-ITSO1and define an exclusive logical drive-to-LUN mappings for the LUNs data1 and metadata1(one storage partition). Then, select Host Group GPFS-ITSO2 and define logical drive-to-LUN mappings for LUNs data2 and metadata2 (a second storage partition).

   Figure 3-8 on page 96 shows the mappings in the Storage Manager GUI.

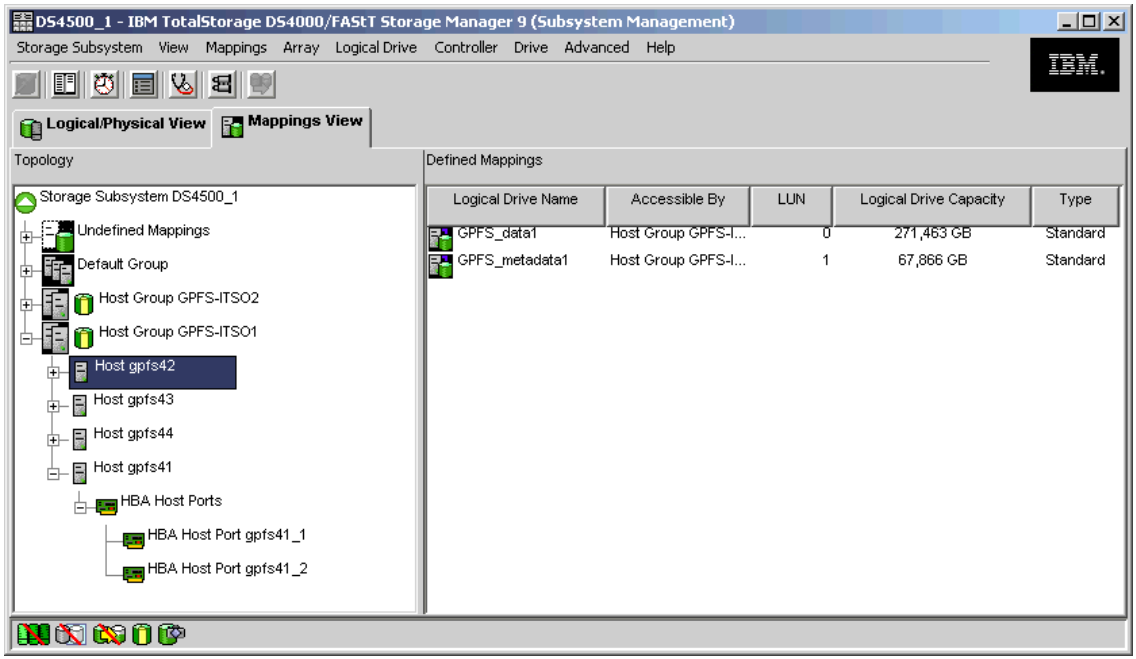

Figure 3-8 DS4500 Storage Manager GUI showing the storage mappings (partitioning)

Storage partitioning (LUN masking) is necessary to prevent other systems outside the cluster to access and configure the LUNs. This is a storage security feature.

3.2.3 Storage Area Network (SAN)

In our environment the SAN consists of two IBM 2109-F32 SAN switches used to to attach multiple hosts to the same storage (using fibre channel cabling and protocol). We use two SAN switches for redundancy and high availability. Both switches are configured the same way, therefore in Example 3-4 we present the devices attached to the first switch.

Tip: If you have a heterogeneous environment, with SAN switches from different vendors, you have to make sure that you configure all SAN elements in compatibility mode for proper interoperability.

Example 3-4 ITSO FC-switch (2109-F32) itsosan1 configuration

```
itsosan01:root> switchShow
switchName:     itsosan01
switchType:     12.4
```

```
switchState:    Online
switchMode:     Native
switchRole:     Principal
switchDomain:   101
switchId:       fffc65
switchWwn:      10:00:00:60:69:90:61:58
zoning:         ON (ITSO_zone_cfg)
switchBeacon:   OFF

Port Media Speed State
=========================
  0   id    N2   Online    F-Port  20:02:00:a0:b8:12:10:6f
  1   id    N2   Online    F-Port  20:03:00:a0:b8:12:10:6f
  2   id    N2   Online    F-Port  10:00:00:00:c9:31:b0:4c
  3   id    N2   Online    F-Port  50:05:07:64:01:40:5c:1f
  4   id    N2   Online    F-Port  10:00:00:00:c9:33:01:8a
  5   id    N2   Online    F-Port  10:00:00:00:c9:42:1e:83
  6   id    N2   Online    F-Port  10:00:00:00:c9:42:2f:78
  7   id    N2   Online    F-Port  10:00:00:00:c9:40:77:58
  8   id    N2   Online    F-Port  10:00:00:00:c9:40:3b:41
  9   id    N2   No_Light
 10   id    N2   No_Light
 11   id    N2   No_Light
 12   id    N2   No_Light
 13   id    N2   No_Light
.....................>>> Omitted lines <<<.................
 30   id    N2   No_Light
 31   id    N2   No_Light
```

The previous Example 3-4 on page 96 you can see the status of each port, the speed settings, and the attached device (identified by its WWN).

Attention: Without zoning, all attached devices could see every other attached device. This poses the following threat:

If a system connected to the SAN is powered up, the HBA device driver will scan the entire SAN looking for LUNs to attach. This happens regardless the status od the LUN masking (which is equivalent with an ACL at storage subsystem level).

The scanning process may interfere with the active systems' operation, and may cause these systems to be disconnected from their LUNs (this is very much dependent on the device driver implementation for each platform).

For every HBA we have created a separate zone, containing two members, the HBA and the storage subsystem controller. Example 3-5 on page 98 shows the

zoning configuration of one switch (itsosan1). As all nodes have two HBAs, the zoning of the second switch is similar, using the second HBA and the second controller (B) of the DS4500 instead. Aliases are used for configuration, as names are easier to remember than numbers.

Example 3-5 FC-switch (2109-F32) itsosani1 profile - partly (zone configuration only)

```
[Zoning]
cfg.ITSO_zone_cfg:GPFS_ITSO1_gpfs41A;GPFS_ITSO1_gpfs42A;GPFS_ITSO1_gpfs43A;GPFS
_ITSO1_gpfs44A;GPFS_ITSO2_gpfs29A;GPFS_ITSO2_gpfs29B;GPFS_ITSO2_gpfs30A;GPFS_IT
SO2_gpfs30B
zone.GPFS_ITSO1_gpfs41A:DS4500_A1;gpfs41_1
zone.GPFS_ITSO1_gpfs42A:DS4500_A1;gpfs42_1
zone.GPFS_ITSO1_gpfs43A:DS4500_A1;gpfs43_1
zone.GPFS_ITSO1_gpfs44A:DS4500_A1;gpfs44_1
zone.GPFS_ITSO2_gpfs29A:DS4500_A1;gpfs29_1
zone.GPFS_ITSO2_gpfs29B:DS4500_B1;gpfs29_2
zone.GPFS_ITSO2_gpfs30A:DS4500_A1;gpfs30_1
zone.GPFS_ITSO2_gpfs30B:DS4500_B1;gpfs30_2
alias.DS4500_A1:20:02:00:a0:b8:12:10:6f
alias.DS4500_B1:20:03:00:a0:b8:12:10:6f
alias.gpfs29_1:10:00:00:00:c9:31:b0:4c
alias.gpfs30_1:10:00:00:00:c9:33:01:8a
alias.gpfs41_1:10:00:00:00:c9:42:1e:83
alias.gpfs42_1:10:00:00:00:c9:42:2f:78
alias.gpfs43_1:10:00:00:00:c9:40:77:58
alias.gpfs44_1:10:00:00:00:c9:40:3b:41
enable:ITSO_zone_cfg
```

3.2.4 Servers

IBM offers a wide range of servers suitable for running GPFS. The latest supported server hardware list can be found at:

http://www.ibm.com/servers/eserver/clusters/software/gpfs.html

Whenever you chose server, make sure it fits your performance and connectivity needs, and also check for latest microcode updates available. Make sure to use the correct files for your platform. Follow the instructions described in the README file.

In our environment we have used several platforms, but we have closed as an example two IBM @server® pSeries 550 with the following configuration:

- 4 processors
- 8 GB RAM
- 8 x 73 GB internal SCSI disks divided in two SCSI buses

- One dual port on-board Gigabit ethernet
- One dual port PCI gigabit ethernet adapter
- Four FC 2765 HBAs (Emulex)

We partitioned the servers Logical Partitioning (LPAR) in two independent logical partitions, each containing half of the resources mentioned above. We have use static partitions only, with dedicated resources.

We have defined the available IDE (CD-ROM) and the USB bus to be a “desired” resource, therefore it can be alternately allocated to each partition (for software installation and updates). The final configuration contains is results in four independent hosts.

Restriction: Keep in mind that without a PCI expansion drawer, the available number of PCI slots of an p5 550 is limited to five.

Figure 3-9 shows the memory allocation for a partition.

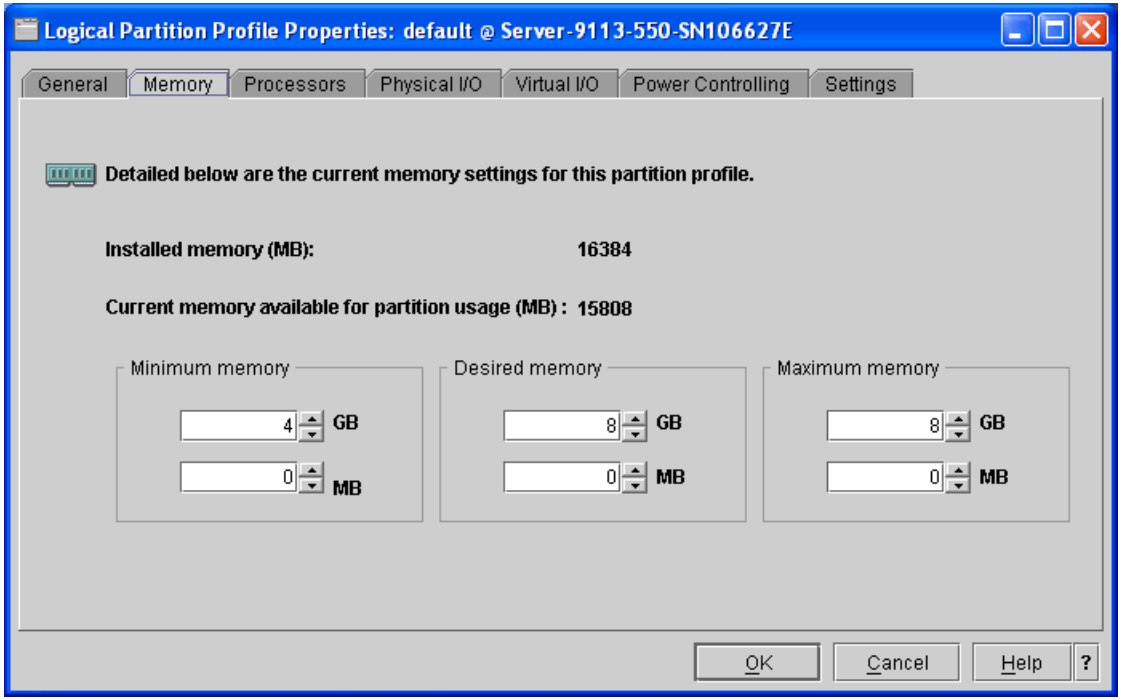

Figure 3-9 LPAR profile of p5 550 - memory configuration

Figure 3-10 on page 100 shows the I/O resource allocation for our systems.

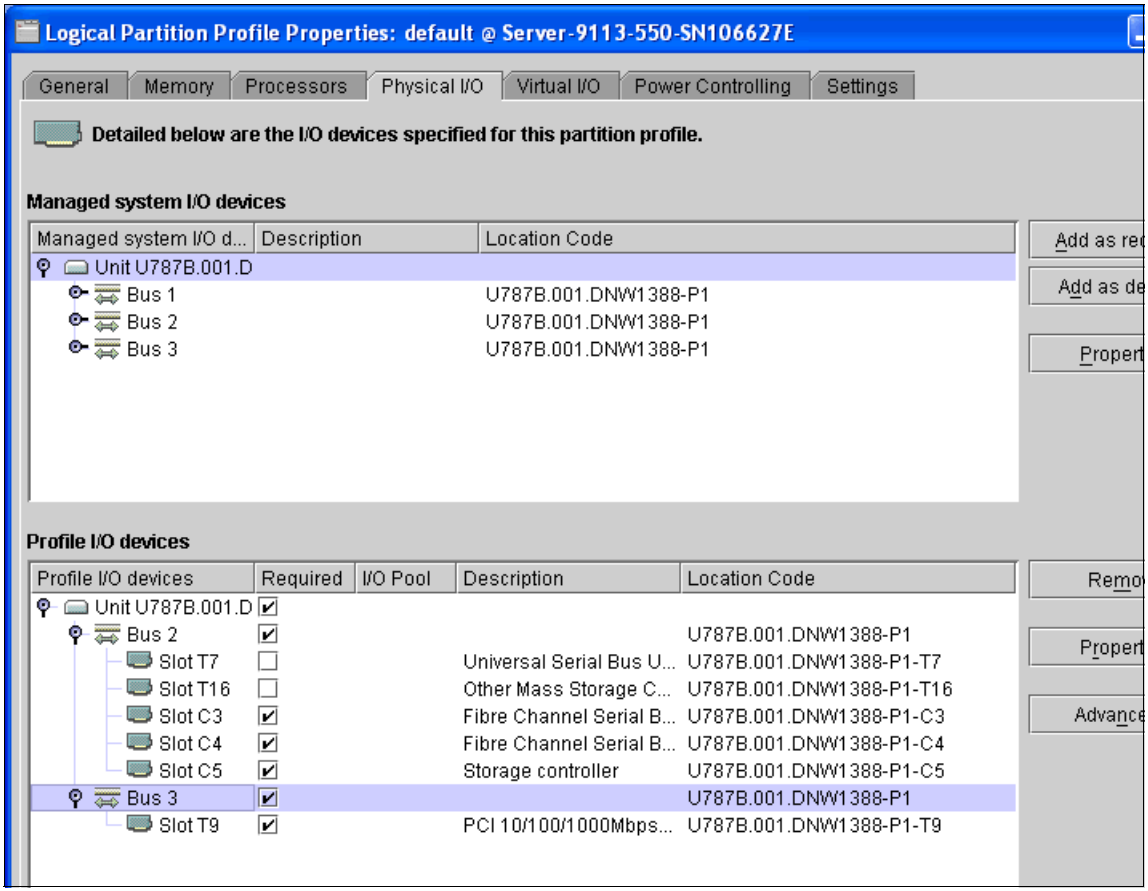

Figure 3-10 LPAR profile of p5 550 - physical I/O configuration

Finally, we have allocated two dedicated processors for each LPAR, as shown in Figure 3-11 on page 101.

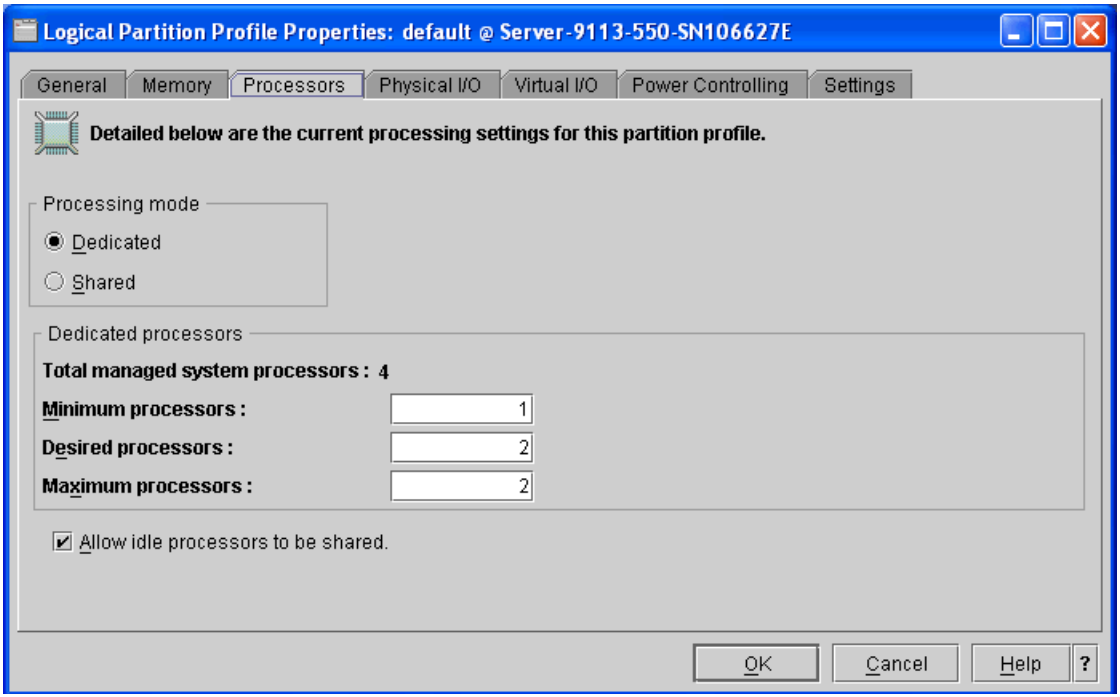

Figure 3-11 LPAR profile of p5 550 - processor configuration

After finishing these steps, the basic configuration is prepared for operating system installation. This will be described in more detail in 3.3, “Software installation (Linux-based cluster)” on page 103.

3.2.5 Network infrastructure

Switches

It is highly recommended that you use non-blocking network switches. This means that all switch ports can sustain the maximum throughput to any other port at all times. Plan your network carefully, as later changes may be difficult to implement or even cause system interruptions. We implemented two different networks (two class C subnets - one for management purposes, the other one dedicated to GPFS).

Subnetting

The first network is used for management activities, like nodes installation, administrative commands, managing the DS4500 and the SAN switches. This network uses the IP address range 192.168.100/24.

The second network will be used as the GPFS cluster communication network, and is a Gigabit ethernet. It uses the IP address range 10.10.100/24.

Jumbo frames

For our environment we have chosen not to use jumbo frames (which allow for larger IP packets) as the normal Gigabit ethernet connection is sufficient.

Note: You may consider using jumbo frames (MTU > 1500 bytes) if you experience heavy traffic with large data blocks, therefore having less data fragmentation.

In certain cases, jumbo frames can improve network performance. The Maximum Transmission Unit (MTU) is the largest size of IP datagram which may be transferred using a specific data link connection. As the normal Ethernet MTU size 1500 bytes, with jumbo frames you can increase this size up to 16384 bytes. Keep in mind that all devices connected to this network must support jumbo frames.

VLAN

VLAN is another feature to increase performance and to avoid troubles as VLANs separate data traffic to devices only belonging to one VLAN.

Link aggregation

Network performance and availability can be further improved by implementing a link aggregation mechanism (Etherchannel, trunking, IEEE 802.3ad) for multiple adapters from nodes to switch. The switch and the network interface cards (NICs) drivers must support the same standard. Linux provides a software link aggregation named “bonding” (described later in “Network link aggregation (optional)” on page 120). For more information about Linux Ethernet bonding, see the Linux Kernel documentation.

3.2.6 ITSO lab environment overview

A diagram of the complete environment we used in our ITSO lab is shown in Figure 3-12 on page 103.

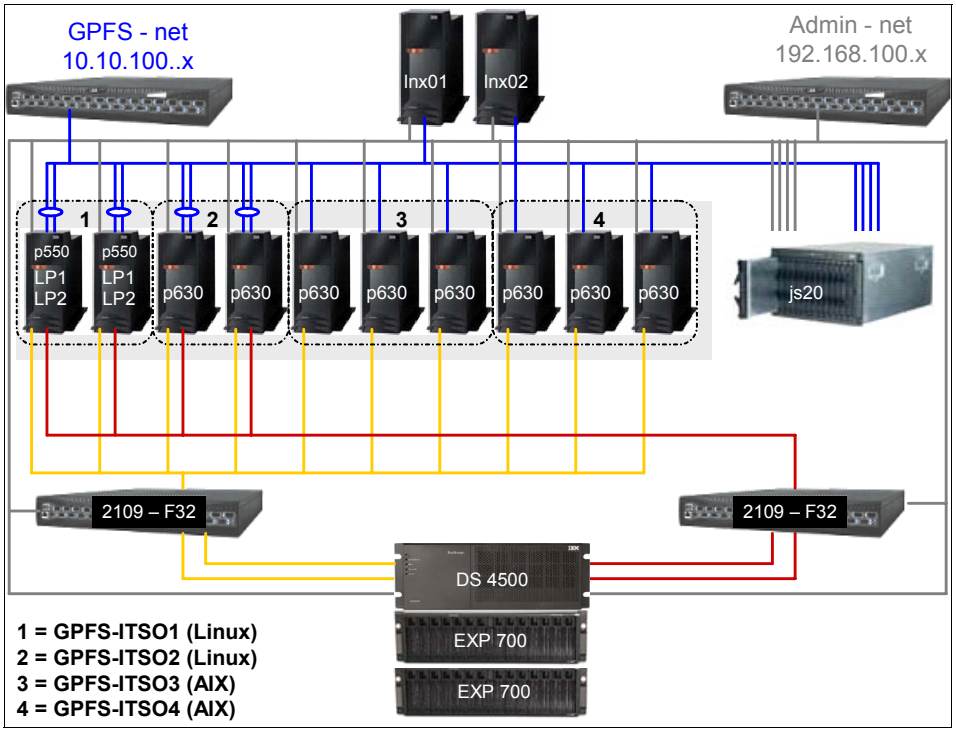

Figure 3-12 ITSO Lab environment overview

Based on these components we have built several GPFS clusters:

- GPFS-ITSO1: four p550 LPARs (running Linux).
- GPFS-ITSO2: three rackmounted p630 (running AIX)
- GPFS-ITSO3: three rackmounted p630 (running AIX)
- GPFS-ITSO4: two standalone p630 (running Linux)

As the IBM @server JS20 blades are currently only supported as GPFS NSD clients, we have not configured any storage attached to them, and we have used them as generic clients (workload generators) for the other GPFS clusters.

3.3 Software installation (Linux-based cluster)

This section covers Linux operating system configuration on IBM @server pSeries systems. As Linux is becoming more popular on pSeries hardware, we consider useful to present a full Linux system installation (on pSeries servers) in this book. For AIX base operating system installation, refer to existing manuals.

Once the Linux operating system has been installed and the storage prepared, GPFS cluster configuration is similar on both AIX and Linux.

Before you proceed to operating system installation, it is mandatory to check the latest support information for compatibility matrixes and microcode updates. Also it is recommended to prepare the following information in advance:

- Naming conventions (for hosts, zones, cluster)
- Designated management station (central point to administer the environment)
- Networking topology (SAN, LAN, GPFS cross cluster)
- Configuration files and/or administrative scripts

3.3.1 Base operating system setup and configuration

After checking the hardware and software prerequisites, install Linux on all nodes. This task is described in detail in several other Redbooks, therefore we show only the major steps here.

SLES9 Installation on an IBM @server pSeries 550

As an example, here we describe how to set up SLES9 on an IBM @server pSeries 550.

Insert the SLES9 CD1 in the CD-ROM drive and start the first logical partition. After the boot process is finished and the installation ramdisk is loaded into the memory, the SUSE Yast installation screen language selection screen appears.

Continue on to the “Installation Settings” screen shown in Figure 3-13 on page 105. For new system “New installation” is selected by default, then the keyboard layout has to be specified. In our case the suggestion for partitioning was acceptable. In case you reinstall an already configured system, we recommend to delete the already configured partitions and create new ones.

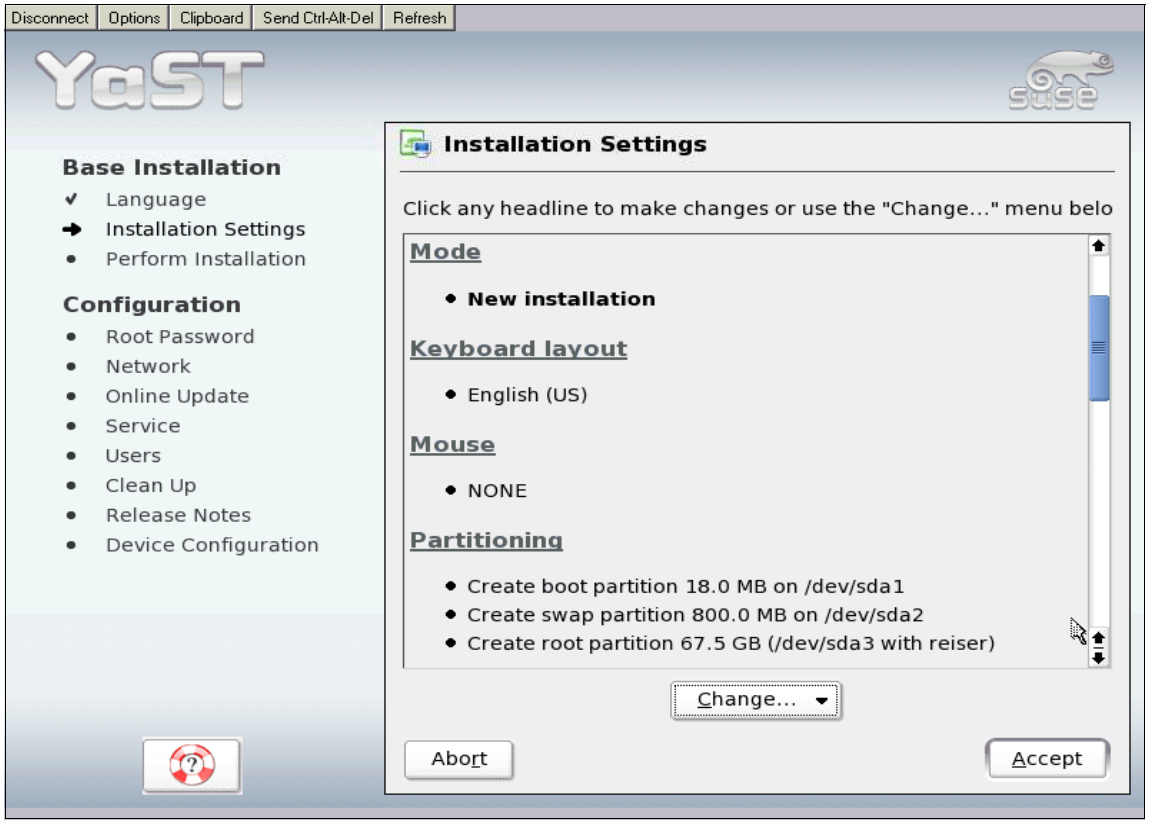

Figure 3-13 SLES9-yast-setup - Installations settings part 1 (mode, keyboard layout, mouse, partitioning)

One of the most important steps is to choose the software packages. As we need to compile the GPFS open source compatibility layer, we need to make sure that the kernel sources and the compiler packages will be installed at least on one machine. We need to install also the XFree86-devel package which contains the **`imake`** utility required to compile the GPFS open source compatibility layer (see Figure 3-14 on page 106).

Figure 3-14 SLES9-yast-setup - Installations settings part 2 (software)

Finally configure the settings for the time zone, language and the default runtime level. As the hosts are used only as computing or file serving nodes, and there is no graphics adapter installed, the default runlevel can be set to 3 (see Figure 3-15 on page 107).

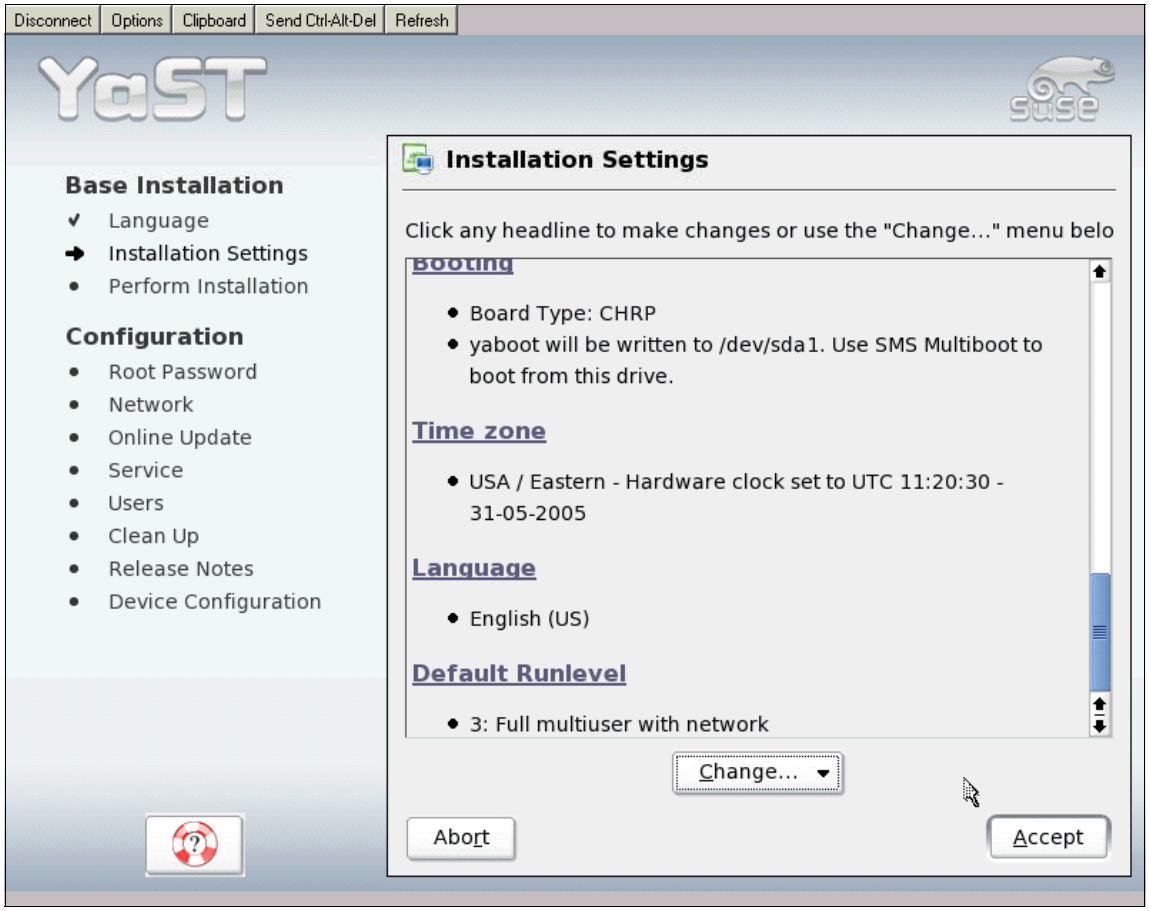

Figure 3-15 SLES9-yast-setup - Installations settings part 3 (booting, time zone, language, runlevel)

Proceed to installation by clicking on the “Accept” button and “Yes, Install” in the next screen (shown in Figure 3-16 on page 108).

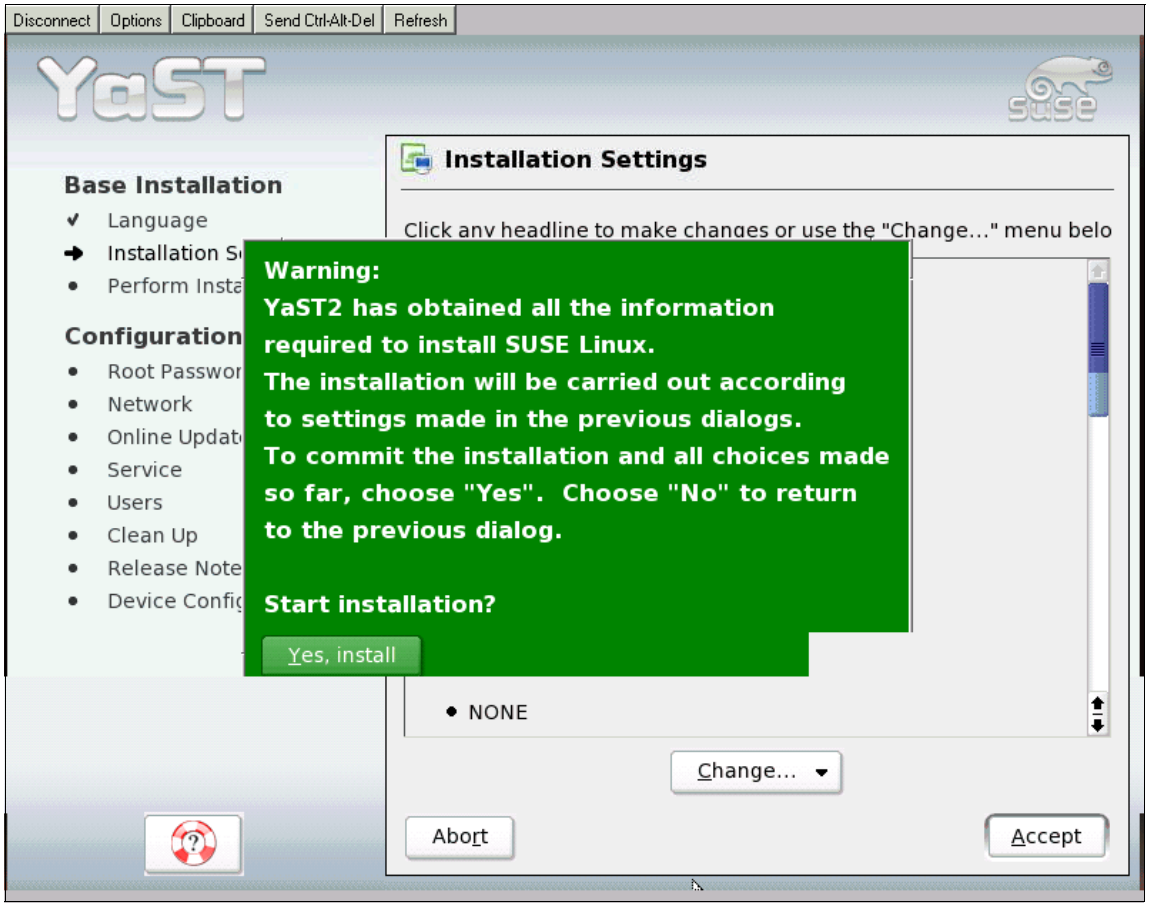

Figure 3-16 Final installation confirmation screen

After pressing the “Yes, install” button, Yast will format the disk, copy the required packages, and finally reboot. The base installation part is finished.

Important: During the operating system installation we strongly recommend to unplug the HBA cables, or to disable the corresponding ports on the FC switches for these connections (check “HBA driver configuration” on page 109 for more details).

This is due to the fact that at reboot time, the HBA adapter may be detected by the hardware scan before the controller attaching the internal boot disk, therefore the disk order and naming will change, and the system will not find the boot device (which has just been installed).

After the system comes up again, final configuration needs to be done. This includes the following steps:

- Choose a password for root user
  You may want to change the password encryption method from DES to MD5, as DES encrypted password is limited to eight characters. You can also change this later by modifying the encryption method in /etc/default/passwd file.
- Add an additional user
  It is generally recommended to add one or more a common users, as it is generally not recommended to log on to the system as “root” user over the network.
- Various services configuration (we skipped the printer, LDAP and test internet connection sections)
- Configure the network
  Enter the hostname and configure the settings related to your environment. We have chosen a static IP address and entered the default gateway, domain name, and name server IP address.

Network configuration concludes the base installation process. The system is now ready for further configuration and customization to bring the system ready for GPFS. These next steps include:

- Installation and configuration of required storage devices, agents or GUIs.
  - Device driver
  - Multipathing software (if applicable)
  - DS4500 client software (if necessary)
- Configure the network
  - Name-to-IP address resolution
  - Secure shell (remote command execution)
  - Distributed shell (optional)
  - Configuration file distribution with **`rdist`** (optional)
  - Time synchronization (highly recommended)
  - File sharing services, like SAMBA or NFS

HBA driver configuration

SLES9 comes by default with the required device drivers. During the base operating system installation, the hardware is detected automatically and the appropriate device drivers are added to the configuration (storage device drivers are added to the initial ramdisk). This can lead to subsequent boot issues.

For example, if storage (LUNs, masking, zoning) is already configured and attached to the host, the device name for the boot disk may change, depending on the PCI bus scanning sequence, and how the internal SCSI adapter and the additional FC HBA(s) are discovered. In case the FC HBA is discovered first, the system will not boot up correctly because the drive order changes.

Attention: The sequence in which the SCSI adapter and the PCI FC HBAs are discovered during system boot is important. The LUNs in the DS4500 may change the disk order/sequence on the system and may lead to boot issues if the HBA driver is included into the initial ramdisk (initrd).

An easy way to avoid troubles is to change the */etc/sysconfig/kernel* file. You should take the Emulex device driver (HBA) out of the initial ramdisk and configure to only be loaded as a module during the normal boot sequence. Example 3-6 shows the lines we changed.

Example 3-6 Changing two lines in /etc/sysconfig/kernel to avoid boot issues

```
.......... >> Omitted lines << ...........
INITRD_MODULES="ipr"
.......... >> Omitted lines << ...........
MODULES_LOADED_ON_BOOT="lpfcdd"
.......... >> Omitted lines << ...........
```

3.3.2 Linux storage device multipath configuration

Linux multipathing is relatively new, and since an unified, or a vendor implemented solution has not been available until recently, this has caused different developers to come up with their own implementations as a temporary solution.

The first implementation was a HBA driver multipathing solution, then RDAC was implemented, but it took some time for RDAC to be considered a reliable solution. Meanwhile the kernel developers implemented a vendor-agnostic multipathing solution as part of the device-mapper subsystem (kernel 2.6).

Currently there are 3 different Linux multipath solutions:

- Vendor-specific HBA driver implementation (Qlogic, Emulex, etc.)
- RDAC multipath driver (from storage manufacturer - here IBM)
- Device-mapper multipath driver (Linux “native” implementation)

As we are using a combination of Linux on POWER™ and IBM DS4500, we have chosen to test the IBM RDAC driver.

The other two multipathing methods are presented in detail in 4.1, “Linux multipathing” on page 140

RDAC/MPP setup and configuration

In case the host has more than one HBA and all HBAs will have access to one or more DS4xxx, a multi path driver is required. For the DS4xxx family this is the RDAC/MPP (redundant disk array controller / multi path proxy) driver. On AIX, the base operating system already contains the RDAC driver (also known as MPIO). For Linux, the RDAC/MPP package must be downloaded and installed manually.

To obtain the RDAC driver you should follow these steps:

- Using a Web browser, open the following URL:

  `http://www.ibm.com/servers/storage/support/disk/`

- This page offers support for disk storage systems. Click the name of the appropriate DS4000 series system.
- This opens a support page for the specific DS4000 series system.
- On the Web page that you have opened, click the Download tab. This shows the fixes and drivers that are available for download.
- In the list of fixes and drivers, click the link for Storage Manager, firmware, HBA and tools (including readmes). This opens a page for Storage Manager, firmware, HBA, and tool downloads for the DS4000 series system.
- On the Web page that you have opened, click the Storage Mgr tab. This shows the downloadable files for the IBM DS4000 Storage Manager.
- Click the link for the IBM TotalStorage DS4000 Linux RDAC (for the Linux Kernel Version 2.4 or 2.6 as appropriate). This opens the download page for the RDAC driver. We used this Web site (this may change):

  `http://www.ibm.com/pc/support/site.wss/document.do?lndocid=MIGR-6O612`

Note: At the time this book is written, Linux RDAC support for kernel 2.6 is not available for Intel® or AMD processor platforms. Check the latest information available.

The RDAC multipath driver consists of two modules:

- **mppUpper**
  This module is loaded before any HBA and prevents the HBA to publicize LUNs to the system.
- **mppVhba**
  This module is a virtual HBA on top of the real HBA. It is responsible for bundling the different paths to the same LUN, organizing the traffic from/to the LUN, and presenting it to the operating system as single entity.

Important: If RDAC/MPP is used, a non-failover HBA device driver is required and AVT (auto volume transfer) MUST be disabled at the DS4xxx level. This is achieved by changing the host type from “Linux” to “LNXCL”. Check the latest version of the Readme.txt delivered with the RDAC/MPP driver and follow the instructions.

The correct host type needs to be chosen, which is in this case LNXCL. Figure 3-17 shows the SM GUI menus used to change the host type for a specific host port. You can also highlight the host port, right-click and choose from the pop-up menu “Change Host Type”.

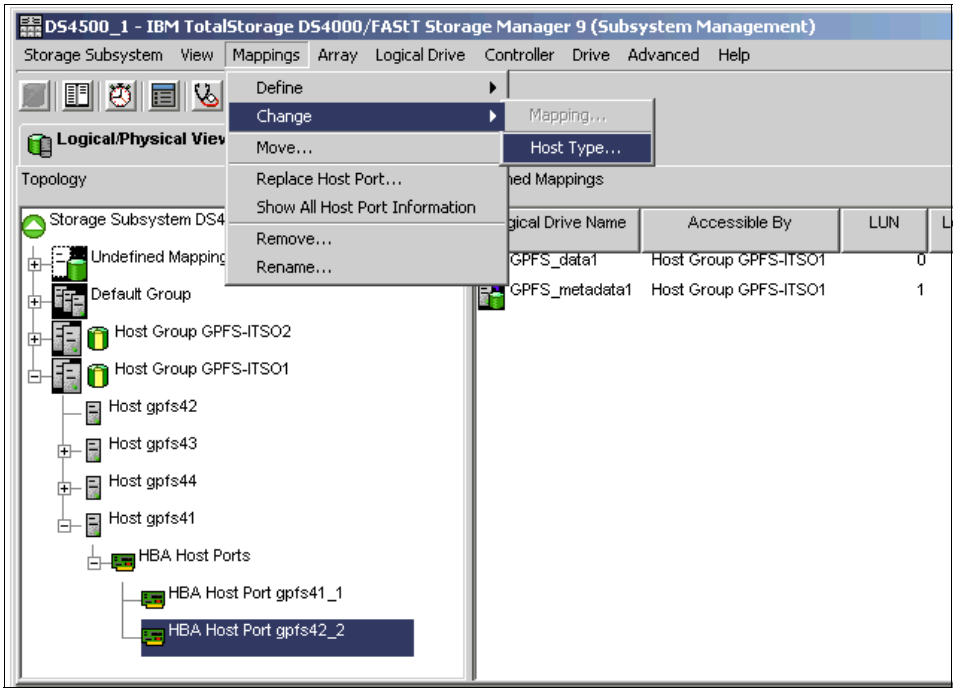

Figure 3-17 DS4500 - menu steps to change the Host Type via the GUI

A new window will appear (see Figure 3-18 on page 113). It shows information about the chosen host port (name, identifier, and current host type) and offers to change the current host type.

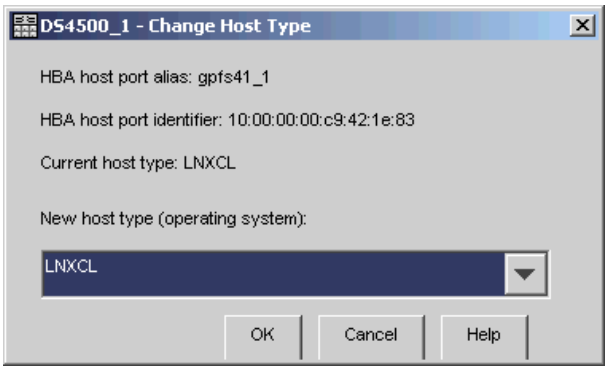

Figure 3-18 DS4500 - Change Host Type

Note: If Storage Partitioning is disabled, the default host type is presented in this field and cannot be changed.

Limitations for RDAC driver on Linux

- The Linux RDAC driver cannot coexist with a Fibre Channel HBA-level multipath failover/failback driver.
- Auto Logical Drive Transfer (ADT/AVT) mode is not supported at this time.
- The Linux SCSI layer does not support skipped (sparse) LUNs.
- The virtual HBA driver cannot be loaded if there is no storage array attached.
- RDAC for Linux kernel 2.6 is currently does not support Linux on Intel or AMD processors (only on POWER systems).
- Clustering is currently not supported on Linux on POWER (LoP) servers.
- (For LoP Servers) the "modprobe" command is not supported with this Linux RDAC release on SLES 9 when the Linux RDAC driver is installed. Use "insmod" to remove and recover the device driver stack.

Attention: Check always the latest version of the Readme.txt that comes with the current RDAC driver, and also the related support matrixes.

Installation and configuration of the RDAC driver

For configuring RDAC, follow these steps:

- Edit the `/etc/sysconfig/kernel` file and make sure the HBA driver is part of the initial ramdisk. In our case, the INITRD_MODULES line will have the following entries:

```
INITRD_MODULES="ipr lpfcdd"
```

where “ipr” is the module for the internal SCSI adapter of the p550, and “lpfcdd” is the module needed for the Emulex HBA.

- Check the RDAC Readme.txt for HBA parameters/options needed.
- Change to the directory where package is stored and unpack the RDAC package:

```
tar -zxvf  rdac-LINUX-xx.xx.xx.xx-source.tar.gz
```

- Compile the and install the driver and utilities:

```
# make clean
# make
# make uninstall (optional; if RDAC was already installed before)
# make install
```

- run **lilo**
- Add an entry for the newly created initial ramdisk image to your boot loader configuration file. The following is an example for adding lines to the /etc/yaboot.conf file:

```
image = /boot/vmlinux
    label = MPP
    root = /dev/sdd3
    append = "selinux=0 elevator=cfq panic=5"
    initrd = /boot/mpp-2.6.5-7.139-pseries64.img
```

Note: Keep in mind to change this example according to your environment. The kernel version (highlighted lines in Example 3-7) must match your environment.

- Reboot the system using the new boot menu option. Although is not mandatory to create a separate boot stanza for RDAC divers, this is highly recommended in case you have problems with the RDAC driver.

Example 3-7 /etc/yaboot.conf file configured to use RDAC

```
# header section
partition = 3
timeout = 100
default = MPP
# image section
image = /boot/vmlinux
    label = linux
    root = /dev/sdd3
    append = "selinux=0 elevator=cfq"
    initrd = /boot/initrd
image = /boot/vmlinux
```

```
label = MPP
root = /dev/sdd3
append = "selinux=0 elevator=cfq panic=5"
initrd = /boot/mpp-2.6.5-7.139-pseries64.img
```

- After reboot, verify that the modules `sd_mod`, `sg`, `mpp_Upper`, `HBA module`, and `mpp_Vhba` are loaded, by using **`/sbin/lsmod`** command. The output for our environment is shown in Example 3-8, with highlighted lines for the required modules for RDAC usage.

Example 3-8 Output of lsmod while using RDAC - important modules highlighted

```
p5n1lp1:~ # lsmod
Module                  Size  Used by
evdev                  31416  0
joydev                 31520  0
st                     73464  0
sr_mod                 44508  0
ehci_hcd               63212  0
ohci_hcd               45484  0
ipv6                  477472  31
e1000                 170868  0
usbcore               183644  4 ehci_hcd,ohci_hcd
subfs                  30168  1
dm_mod                110424  0
mppVhba               149616  0
lpfcdd                763072  4
ipr                   104960  2
firmware_class         32384  1 ipr
mppUpper              151616  1 mppVhba
sg                     74432  0
sd_mod                 43792  3
scsi_mod              194384  8 st,sr_mod,mppVhba,lpfcdd,ipr,mppUpper,sg,sd_mod
```

If you have problems using the RDAC multipath drivers in combination with the local SCSI disk drivers, you have to verify the following:

- Make sure the *sd_mod* and *sg* modules are loaded before ant other storage modules (not the *sr* module), otherwise *mppUpper* might fail to load.
- Make sure the local SCSI disk driver is loaded before any other HBA driver, otherwise you might have problems booting your system because the local boot disk name does not match the one in the configuration.
- Make sure the *mppUpper* module is loaded before any other HBA driver (*lpfcdd*, *qla2xxx*), otherwise kernel panics might be encountered.

An easy way to verify that the created initial ramdisk incorporates the previously mentioned rules is to check the entries of the newly created boot image. Example 3-9 shows an example of a bad configuration image.

This will cause trouble because the module loading sequence is wrong as `lpfcdd` is loaded before the internal SCSI driver. This can lead to reordering the devices and will result in failing to boot, as the specified boot device (specified within the bootloader configuration) cannot be found any more.

Example 3-9 Defect boot image- order of modules is incorrect

```
p5n1lp1:~ # zcat /boot/mpp-2.6.5-7.139-pseries64.img | strings | grep '^insmod
/lib'
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/scsi_mod.ko
$extra_scsi_params max_report_luns=256
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/sd_mod.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/sg.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/mppUpper.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/lpfc/lpfcdd.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/base/firmware_class.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/ipr.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/mppVhba.ko
```

Example 3-10 shows a correct module loading sequence.

Example 3-10 Correct boot image

```
p5n1lp1:~ # zcat /boot/mpp-2.6.5-7.139-pseries64.img | strings | grep '^insmod
/lib'
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/scsi_mod.ko
$extra_scsi_params max_report_luns=256
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/sd_mod.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/sg.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/mppUpper.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/base/firmware_class.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/ipr.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/lpfc/lpfcdd.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/mppVhba.ko
```

If you have problems creating the boot image following these rules on SLES9, you can force it to work, using the following command.

```
mkinitrd -m "sd_mod sg mppUpper ipr lpfcdd mppVhba" -i "mpp-$(uname
-r).img" -k "vmlinux"
```

In this example “`ipr`” is the driver for the internal SCSI disks and “`lpfcdd`” is the driver for external FC-attached storage.

RDAC/MPP stores information in the /proc file system. The following information about RDAC can be found in the /proc directory (see also Example 3-11):

- /proc/mpp

  Entry for the RDAC driver itself

- /proc/scsi/mpp/<adaptor number>

  The entry for the MPP virtual host adapter. This can differ from system to system and is assigned by the scsi middle-layer.

- /proc/mpp/<storage-array name>

  There is an entry for each array that is visible to the host.

- /proc/mpp/<storage-array name>/controllerA
  /proc/mpp/<storage-array name>/controllerB

  These are the entries for the two targets on each array seen by the host. Each storage array has two controllers.

- /proc/mpp/<storage-array name>/controller<A/B>/<low-level-driver><HCT#>

  Here the low level driver can be either for Qlogic or Emulex HBAs. The HCT# is the Host#, channel#, Target#, where Host# is the Host number of the low-level driver as assigned by the SCSI mid-layer. Channel# depends on whether the HBA card is single or dual channel. Target# is the number assigned to that controller on that array by the Low-level HBA driver.

- /proc/mpp/<storage-array name>/controller<A/B>/<low-level-driver><HCT#>/LUN #

  These are the Volumes or LUN # of the Volumes as mapped on host partition on the storage array and seen through that path or Initiater(Host)-Target combination.

Example 3-11 Output of ls -lR

```
p5n1lp1:~ # ls -lR /proc/mpp
/proc/mpp:
total 0
dr-xr-xr-x   3 root root 0 Jun  9 11:23 .
dr-xr-xr-x  92 root root 0 Jun  9 11:20 ..
dr-xr-xr-x   4 root root 0 Jun  9 11:23 DS4500_1

/proc/mpp/DS4500_1:
total 0
dr-xr-xr-x  4 root root 0 Jun  9 11:23 .
dr-xr-xr-x  3 root root 0 Jun  9 11:23 ..
dr-xr-xr-x  3 root root 0 Jun  9 11:23 controllerA
dr-xr-xr-x  3 root root 0 Jun  9 11:23 controllerB
-rw-r--r--  1 root root 0 Jun  9 11:23 virtualLun0
-rw-r--r--  1 root root 0 Jun  9 11:23 virtualLun1
```

```
/proc/mpp/DS4500_1/controllerA:
total 0
dr-xr-xr-x  3 root root 0 Jun  9 11:23 .
dr-xr-xr-x  4 root root 0 Jun  9 11:23 ..
dr-xr-xr-x  2 root root 0 Jun  9 11:23 lpfc_h1c0t0

/proc/mpp/DS4500_1/controllerA/lpfc_h1c0t0:
total 0
dr-xr-xr-x  2 root root 0 Jun  9 11:23 .
dr-xr-xr-x  3 root root 0 Jun  9 11:23 ..
-rw-r--r--  1 root root 0 Jun  9 11:23 LUN0
-rw-r--r--  1 root root 0 Jun  9 11:23 LUN1

/proc/mpp/DS4500_1/controllerB:
total 0
dr-xr-xr-x  3 root root 0 Jun  9 11:23 .
dr-xr-xr-x  4 root root 0 Jun  9 11:23 ..
dr-xr-xr-x  2 root root 0 Jun  9 11:23 lpfc_h2c0t0

/proc/mpp/DS4500_1/controllerB/lpfc_h2c0t0:
total 0
dr-xr-xr-x  2 root root 0 Jun  9 11:23 .
dr-xr-xr-x  3 root root 0 Jun  9 11:23 ..
-rw-r--r--  1 root root 0 Jun  9 11:23 LUN0
-rw-r--r--  1 root root 0 Jun  9 11:23 LUN1
```

Another available command (**lsscsi**) can be used to check the relation between SCSI numbering and device name. The output is shown in Example 3-12.

Example 3-12 lsscsi output

```
p5n1lp1:~ # lsscsi
[0:0:3:0]    disk    IBM      IC35L073UCDY10-0 S28G  /dev/sda
[0:0:4:0]    disk    IBM      IC35L073UCDY10-0 S28G  /dev/sdb
[0:0:5:0]    disk    IBM      IC35L073UCDY10-0 S28G  /dev/sdc
[0:0:8:0]    disk    IBM      IC35L073UCDY10-0 S28G  /dev/sdd
[0:0:15:0]   enclosu IBM      VSBPD4E2  U4SCSI 5105  -
[0:255:255:255]no dev  IBM      5702001          0150  -
[1:0:0:0]    disk    IBM      1742-900         0520  -
[1:0:0:1]    disk    IBM      1742-900         0520  -
[2:0:0:0]    disk    IBM      1742-900         0520  -
[2:0:0:1]    disk    IBM      1742-900         0520  -
[3:0:0:0]    disk    IBM      VirtualDisk      0520  /dev/sde
[3:0:0:1]    disk    IBM      VirtualDisk      0520  /dev/sdf
```

The **lsscsi** command displays the following items per line:

```
[scsi adapter id:channel id:scsi id:Lun id] devicetype devicevendor
devicemodel deviceversion devicemapping
```

You can also check the /proc directory by using the command to list all available scsi devices:

```
cat /proc/scsi/scsi
```

The output of this command is shown in Example 3-13.

Example 3-13 cat /proc/scsi/scsi output

```
p5n1lp1:~ # cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 03 Lun: 00
  Vendor: IBM      Model: IC35L073UCDY10-0 Rev: S28G
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 04 Lun: 00
  Vendor: IBM      Model: IC35L073UCDY10-0 Rev: S28G
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 05 Lun: 00
  Vendor: IBM      Model: IC35L073UCDY10-0 Rev: S28G
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 08 Lun: 00
  Vendor: IBM      Model: IC35L073UCDY10-0 Rev: S28G
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 15 Lun: 00
  Vendor: IBM      Model: VSBPD4E2  U4SCSI Rev: 5105
  Type:   Enclosure                        ANSI SCSI revision: 02
Host: scsi0 Channel: 255 Id: 255 Lun: 255
  Vendor: IBM      Model: 5702001          Rev: 0150
  Type:   Unknown                          ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: IBM      Model: 1742-900         Rev: 0520
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 00 Lun: 01
  Vendor: IBM      Model: 1742-900         Rev: 0520
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi2 Channel: 00 Id: 00 Lun: 00
  Vendor: IBM      Model: 1742-900         Rev: 0520
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi2 Channel: 00 Id: 00 Lun: 01
  Vendor: IBM      Model: 1742-900         Rev: 0520
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi3 Channel: 00 Id: 00 Lun: 00
  Vendor: IBM      Model: VirtualDisk      Rev: 0520
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi3 Channel: 00 Id: 00 Lun: 01
  Vendor: IBM      Model: VirtualDisk      Rev: 0520
```

```
  Type:   Direct-Access                    ANSI SCSI revision: 03
```

In Example 3-13 on page 119, the same LUNs (LUN0 and LUN1) are seen over two different paths (each LUN is seen twice), but will be presented to the operating system only once, as `virtualLun0` and `virtualLun1`. This can be seen in Example 3-12 on page 118, where you can see that only the virtualLuns have a device name (`/dev/sde` and `/dev/sdf`) assigned.

For future LUN changes you need to keep in mind the following:

- Adding LUN(s)
  After adding LUNs you need to run the **`mppBusRescan`** command. This will update the device lists without a reboot.
- Deleting LUN(s)
  RDAC does not support LUN deletion. Therefore a reboot is required to update the configuration.

Restriction: The Linux RDAC driver does NOT support LUN deletion. You need to reboot the server after deleting the mapped logical drives.

Reboot the system whenever you need to unload the driver stack. The use of **`modprobe`** or **`rmmod`** commands with the RDAC driver stack is not recommended nor supported.

3.3.3 Basic network configuration

Keep in mind that GPFS needs a cross-cluster communication network. It is a good idea to separate it from the normal communication network. So plan your network carefully to separate into management and/or client access network.

Network link aggregation (optional)

Linux implements a software version of Ethernet link aggregation (a.k.a etherchannel, or trunking), which is called bonding. Bonding supports a lot of Ethernet devices, and does not require specialized network hardware nor much effort for implementation and administration.

You should consider using Ethernet bonding whenever high availability and increase throughput are required for either GPFS or NFS/Samba.

Before configure bonding, check the latest kernel documentation and the distribution documentation. Even though bonding is included with the Linux kernel, the actual configuration may be distribution dependent. Follow these basic steps:

- Add the following line to /etc/modules.conf

```
alias bond0 bonding
```

- Define the `bond` network device (/etc/sysconfig/network/bond#) where # represents an integer for the number of bonding devices

```
DEVICE=bond0
IPADDR=192.168.100.41
NETMASK=255.255.255.0
NETWORK=192.168.100.0
BROADCAST=192.168.100.255
ONBOOT=yes
BOOTPROTO=none
USERCTL=no
```

- Change all network interfaces' configuration files to reflect the fact that they will be bonded (they become "slave" for bond0). Usually the configuration files can be found in /etc/sysconfig/network (SUSE LINUX), or /etc/sysconfig/network-scripts (Red Hat).

```
DEVICE=eth0
USERCTL=no
ONBOOT=yes
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
```

- If your distribution does not support master/slave convention inside the network interface configuration files, you will need to manually configure the bonding device with the following commands (check your distribution documentation):

```
/sbin/ifconfig bond0 192.168.100.41 netmask 255.255.255.0 up
/sbin/ifenslave bond0 eth0
/sbin/ifenslave bond0 eth1
```

- Create a script containing these commands and place it into the appropriate /etc/rc.X directory. (to be executed at startup).
- Restart your network configuration or reboot.

The default bonding policy is "round-robin". In case changes are required, look into the bonding documentation which can be found under the kernel source directory:

```
/usr/src/linux/documentation/networking/bonding.txt
```

IP name resolution

Make sure that the name resolution is configured identical for all hosts in the cluster (including clients) for both direct and reverse resolution. We suggest to use static (flat file) name resolution. DNS is another option, but this could create

problem with the GPFS cluster if DNS is not functioning properly (not available, slow response, etc.). Therefore, you can:

- Add all IP-addresses and related host names into the /etc/hosts file and distribute it to all hosts (UNIX and UNIX-like systems).
- Or, you can set up a DNS server (optional).

Important: Make sure all nodes in the cluster resolve names identically. This is critical for both GPFS and the remote command execution program.

Tip: Use only lower cases and avoid using special characters.

If you are using local name resolution, keep in mind that using underscore ("_") character in the names is not supported in a DNS environment and may create problems in case you want to migrate to DNS.

Example 3-14 Entries of /etc/hosts file (partly shown)

```
127.0.0.1       loopback localhost
192.168.100.41 p5n1lp1
192.168.100.42 p5n1lp2
192.168.100.43 p5n2lp1
192.168.100.44 p5n2lp2
#
# GPFS Network
10.10.100.41    gpfs41
10.10.100.42    gpfs42
10.10.100.43    gpfs43
10.10.100.44    gpfs44
#
```

In our environment we have chosen static name resolution (/etc/hosts file), and also made sure that the name resolution search order points to the static file (see `/etc/nsswitch.conf` file on Linux, and `/etc/netsvc.conf` on AIX).
Example 3-14 shows the entries for the hosts we configured in our test implementation.

3.3.4 Configuring secure shell

Open secure shell (OpenSSH) is becoming the choice for securing remote command execution and remote access in a cluster environment. On Linux secure shell is installed automatically (both client and server).

On AIX you need to download the packages from IBM's AIX Toolbox for Linux site, then install them on all nodes in the cluster. As the packages (both Open SSL and Open SSH) use cryptographic content, you need to register to the IBM Web site before downloading the packages:

http://www.ibm.com/servers/aix/products/aixos/linux/download.html

Tip: We recommend that you designate an administrative node within the GPFS quorum nodes (or one of the nodes you consider most reliable).

One convenient way (we used for our tests) to get an environment in place where all hosts can execute authenticated remote commands without the need for typing a password (user intervention) is described in the following steps (you need to be logged in as user root):

- Create a secure key on the management station by using the **ssh-keygen** command. This will generate for the actual user a public (id_rsa.pub) and a private (id_rsa) key stored in root's `.ssh` directory (~/.ssh).

  ```
  ssh-keygen -t rsa
  ```

- Add the key to the `authorized_keys` file.

  ```
  cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
  ```

- Log in to all cluster nodes (including the management station) over the network (using **ssh**). In case the nodes are connected to multiple networks, repeat this step for every network. This login procedure will add the remote **sshd** keys to the local root's ~/.ssh/known_hosts file. Use in example (while gpfs42 is one node name):

  ```
  ssh gpfs42 date
  ```

- Copy the contents of the ~/.ssh directory of the management station to all nodes of the managed cluster.

  ```
  scp -r /root/.ssh gpfs42:/root
  ```

  Note: In this configuration, a single pair of keys will be used for the root user on all nodes. If this is not acceptable in your environment, you might consider generating the ssh client key pair on each node individually, and then concatenate all public keys in a single ~/.ssh/authorized_keys file, which should be distributed to all nodes in the cluster along with the ~/.ssh/known_hosts from the management workstation.

  One useful command to collect root's public keys from all nodes is **ssh-kescan**.

3.3.5 Time synchronization

Check the time synchronization for cluster nodes. Within a cluster environment it is essential that all nodes have the same time. This is necessary for files and log records time stamps, as well as for other cluster activities. In our cluster we have decided to create a local time server (master), as this is acceptable in most environments. Choose one of your hosts as the “time master” (this may be also connected to an outside NTP server if desired). and assign all other cluster nodes as clients for this master (the master will be the time synchronization source).

Attention: The clock of all nodes in the GPFS cluster must be synchronized, otherwise client access (NFS/SAMBA, other applications) may produce undesired results and inconsistencies in the GPFS file system.

Time server (master) configuration - Linux

On the time master server do the following steps:

- Configure your server's time zone using the **`tzselect`** command. Follow the Time Zone Wizard prompts to set your time zone.
- Check if the required **`xntpd`** daemon is installed:

  `ls /etc/init.d/xntpd.`
- If installed, save the existing `ntp.conf` file.
  `mv -f /etc/ntp.conf /etc/ntp.conf.ORIG`
- Create a new `/etc/ntp.conf`, as shown in Example 3-15:

Example 3-15 ntp.conf (master) file

```
server 127.127.1.0
fudge 127.127.1.0 stratum 10
driftfile /etc/ntp/drift
```

If you have access to an external time server source you can add the appropriate entry before the first line:

Tip: If you have access to an external time server source you can put an entry before the first line of Example 3-15:

```
server IP-address (of external time source)
```

External time servers can be found at:

http://ntp.isc.org/bin/view/Servers/WebHome

Select one of the Public NTP Secondary (stratum 2) Time Servers from the list provided, that is located reasonably close to you (check the network response time before selecting this server).

- Follow these steps to complete the process and activate the time synchronization on your server:

```
date
hwclock
chkconfig -a xntpd
rcxntpd restart
```

- Finally check if the time server is working by entering:

```
ntpdate -q localhost
```

- If successful, you go to the next step.

Note: Depending on the time difference between the servers, it may take some time before the service is working properly. You can repeat the **ntpdate** command until you see the desired results.

Time servant (client) configuration

To configure all other hosts within the cluster to receive the time from the time master, follow these steps:

- Edit the `/etc/ntp.conf` file on all cluster nodes.
- Use the ntp.conf file from Example 3-15 on page 124 and add as the IP address of the time master on the first line.
- Finally verify the time synchronization. The output should look like this:

Example 3-16 Check for time synchronization

```
shmux -c "date" -</tmp/nodes
p5n1lp1: Wed May 18 11:01:56 EDT 2005
p5n1lp2: Wed May 18 11:02:01 EDT 2005
p5n2lp1: Wed May 18 11:01:54 EDT 2005
p5n2lp2: Wed May 18 11:01:56 EDT 2005
```

Windows client time synchronization

If you like to synchronize your windows clients as well you need to do the following steps (if you not already have in your windows environment a server acting as a time server):

- Define one or multiple samba server as time server by adding the following line to the global section of the /etc/samba/smb.conf:

  ```
  time server = yes
  ```

- Finally you need to insert the following command into the logon script of each windows client, where the IP-ADDRESS is one of the just specified Samba servers acting as a time server:

  ```
  net time \\IP-ADDRESS /set /yes
  ```

Note: The Windows user must have the privilege to change the local time.

3.4 GPFS installation and configuration

This section provides information about how to configure a GPFS cluster in an AIX/Linux environment using NSD on top of raw physical disks (LUNs), without using LVM or VSD.

Before you proceed to configuring your GPFS cluster, check the latest GPFS information (including documentation updates and latest supported configurations) at:

http://www.ibm.com/servers/eserver/clusters/software/gpfs.html

The GPFS documentation can be found at:

http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfsbooks.html

GPFS installation in nine steps

In the previous sections in this chapter we described the configuration steps to perform before the actual GPFS configuration. By thoroughly preparing the environment the GPFS installation will be seamless and can be done within minutes. As a reminder, we present again an overview of the steps needed for a successful GPFS cluster implementation:

1. Install the nodes (operating system, additional software)
2. Configure the networking (switches, connectivity, IP)
3. Verify node-to-node remote command execution and time synchronization
4. Attach servers nodes to storage and verify

5. Install GPFS packages and define a GPFS cluster
6. Start GPFS and verify all nodes join the cluster
7. Create VSDs/NSDs
8. Create file systems
9. Mount file systems

All steps mentioned prior to the actual GPFS configuration must be completed and must be working properly in order to provide GPFS with a properly operating infrastructure. Failure to complete the prior steps without errors will result in a more complicated problem determination. As the steps 1 through 4 are all described in the previous sections, we start this section with step 5.

3.4.1 Install GPFS packages and define a GPFS cluster

Install packages

Install the base GPFS package which contains the license agreement. Then, if necessary, apply the latest PTFs. These can be downloaded from:

```
http://techsupport.services.ibm.com/server/gpfs
```

Note: You need to download and install the packages (PTFs) for your platform (AIX, Linux - x86, PPC, IA64, x86-64, etc.).

Follow the instructions described in the following paragraphs (also consult the *General Parallel File System (GPFS) for Clusters: Concepts, Planning, and Installation*, GA22-7968, and the PTF readme files).

Configuring the GPFS cluster

After the packages are installed you need to compile the Linux portability layer. This is documented in detail the README located in the `/usr/lpp/mmfs/src` directory:

```
cd /usr/lpp/mmfs/src/config
cp site.mcr.proto site.mcr
```

- Edit the `site.mcr` file and change following parameters to according to your environment (we are using SLES9 on PPC; see Example 3-17 on page 128, the highlighted lines):

```
#define GPFS_ARCH_your-platform (here: PPC64)
LINUX_DISTRIBUTION=your-distribution (here: SUSE_LINUX)
LINUX_DISTRIBUTION_LEVEL (here: 9)
#define LINUX_KERNEL_VERSION (here: 2060507)
KERNEL_HEADER_DIR = (here: /lib/modules/‘uname -r‘/source/include)
```

- Save the changes to `site.mcr` file, then:

```
cd /usr/lpp/mmfs/src
export SHARKCLONEROOT=/usr/lpp/mmfs/src
make World
make InstallImages
```

- Check if the new modules have been created:

```
ls -ltr /usr/lpp/mmfs/bin (see the last 5 lines in the command output)
```

Example 3-17 Customized site.mcr file

```
/* Copyright IBM 1995,2003 */
/* $Id: site.mcr.proto,v 1.496.2.5 2005/04/19 22:55:20 desanch Exp $ */

/* Platform and architecture defines.  Exactly one platform (operating
 * system) and architecture (instruction set) must be uncommented.
 * Not all combinations are supported.
 */

/* #define GPFS_AIX */
#define GPFS_LINUX

/* #define GPFS_ARCH_POWER */
/* #define GPFS_ARCH_I386 */
/* #define GPFS_ARCH_IA64 */
#define GPFS_ARCH_PPC64
/* #define GPFS_ARCH_X86_64 */

#ifdef GPFS_LINUX
/****************************************************/
/* Linux portability layer defines begin.           */
/* Customer editable portion of config file.        */
/****************************************************/

/* Linux distribution (select/uncomment only one)
 *
 * Select REDHAT_AS_LINUX for:
 *   - RedHat Advanced Server
 *   - RedHat Enterprise Linux (AS|ES|WS).
 *
 * Select REDHAT_LINUX for RedHat professional
 *
 * Select SUSE_LINUX for SuSE Enterprise Server
 */
/* LINUX_DISTRIBUTION = REDHAT_LINUX */
/* LINUX_DISTRIBUTION = REDHAT_AS_LINUX */
LINUX_DISTRIBUTION = SUSE_LINUX
/* LINUX_DISTRIBUTION = KERNEL_ORG_LINUX */
```

```
/* The DISTRIBUTION_LEVEL may be required if the patches
 * for a common LINUX_KERNEL_VERSION diverge across a
 * LINUX_DISTRIBUTION.  For instance, RedHat 8 and RedHat 9
 * have different kernel patches for a similar version
 * number kernel.
 */
#define LINUX_DISTRIBUTION_LEVEL 9

/* Linux kernel version examples.
 *
 *   2.4.18-21 would be 2041821
 *
 * You may need to explicilty define and/or
 * uncomment your particular level.
 * Note: if the last component of the version
 * number is larger than 100, specify 99 instead,
 * e.g.
 *   2.4.21-111 would be 2042199
 *
 * Version numbers should only include the first
 * 4 components
 * e.g.
 *   2.6.5-7.139 would be 2060507
 */
#define LINUX_KERNEL_VERSION 2060507
/* #define LINUX_KERNEL_VERSION 2042028 */
/* #define LINUX_KERNEL_VERSION 2041814 */
/* #define LINUX_KERNEL_VERSION 2041800 */
/* #define LINUX_KERNEL_VERSION 2040900 */


/* Linux kernel header dir.  This is where GPFS
 * will find the kernel headers for the kernel
 * listed in LINUX_KERNEL_VERSION.
 *
 * Note on SLES9 the default path must be changed.
 */
/* KERNEL_HEADER_DIR = /lib/modules/`uname -r`/build/include */
KERNEL_HEADER_DIR = /lib/modules/`uname -r`/source/include
/* KERNEL_HEADER_DIR = /usr/src/linux/include */

/* Flags for building with desired Linux kernel patches.
 * These are patches for specific problems we've discovered
 * with the linux kernel and are providing interim patches
 * for.  Each patch is described below and the desired patches
 * should be uncommented under LINUX_PATCH_DEFINES.
 *
 * -DMMAP_LINUX_PATCH  Required for use of mmapped writes.
 *
```

```
 * Descriptions and downloads for these patches may be
 * found at http://www-124.ibm.com/linux/patches/?project_id=132
 */
LINUX_PATCH_DEFINES = \
/*  -DMMAP_LINUX_PATCH  */

/****************************************************/
/* Linux portability layer defines end.             */
/* Everything below here is irrelevent to customer  */
/* compiles of the portability layer.               */
/****************************************************/
#endif /* GPFS_LINUX */

..........>>> Omitted lines - no further changes to this file <<<..........
```

Tip: The site.mcr is a C source file, therefore the comments start with "/*" and end with a "*/".

GPFS directories and files

After GPFS installation, the structure looks like this:

- /usr/lpp/mmfs - GPFS installation directory
  - `/usr/lpp/mmfs/bin`
    Location for GPFS executables (scripts and binaries)

    Note: The user commands start with "mm" (Multi-Media), while the compiled binaries (NOT directly executable by users) start with "ts" (Tiger Shark).

  - `/usr/lpp/mmfs/src`
    Location of the GPFS Portability Layer Source (Linux only)
- `/var/mmfs` - GPFS config data directory
  - `/var/mmfs/gen`
    Location for critical GPFS config data
  - `/var/mmfs/etc`
    Location for GPFS specific user scripts and cluster specific custom config files (mmfs.cfg and cluster.preferences)
- `/etc/cluster.nodes` – Tracks node IP addresses and host names

Now GPFS is ready to use and we proceed to GPFS cluster configuration. We have created two descriptor files:

- A node descriptor file (see Example 3-18)
- A disk descriptor file (see Example 3-19 on page 132)

The node descriptor file contains all node names or IP-addresses, and the role the host will play within the cluster. Possible options are:

- Manager or client
  Indicates whether a node is part of the pool of nodes from which configuration and file system managers are selected. The default is client.
- Quorum or non-quorum
  Indicates whether a node is counted as a quorum node. The default is non-quorum.

A sample node descriptor file is shown in Example 3-18. The nodes (gpfs41, gpfs42, and gpfs43) belong to the pool of quorum nodes. We have chosen a node quorum method for our installation. As the quorum rule is defined as - half of the number of quorum nodes plus one - at least two of these nodes need to be active to keep the cluster alive.

Note: GPFS 2.3 offers another cluster quorum method - using tiebreaker disks. This method uses only two nodes as quorum nodes (although more quorum nodes can be defined if desired), and up to three NSDs physically attached to both quorum nodes. This is the preferred quorum method in a two node cluster.

Two of these nodes (gpfs41, and gpfs43) belong to the pool of manager nodes and can act as a configuration and/or file system manager node. Node gpfs44 has no further designations defined and therefore is by default a non-quorum node acting as a client.

Example 3-18 Sample GPFS node descriptor file

```
gpfs41:quorum-manager
gpfs42:quorum
gpfs43:quorum-manager
gpfs44
```

Important: Choose your node designations carefully as it can impact the overall system performance and availability. If your application is computing intensive, you might consider not to run applications on top of a file system or configuration manager node as these two functions may be affected, therefore impacting the whole cluster.

The disk descriptor file contains all disks, defines their usage, and specifies to which failure group the disk will belong. A disk descriptor is defined as:

```
DiskName:PrimaryServer:BackupServer:DiskUsage:FailureGroup:DesiredName
```

Primary and Backup servers are specified in case these servers act as NSD servers for this disk. Possible options for “DiskUsage” are (for further details see *General Parallel File System (GPFS) for Clusters: Administration and Programming Reference*, SA22-7967):

- **dataAndMetadata**
  Indicates that the disk contains both data and metadata. This is the default.
- **dataOnly**
  Indicates that the disk contains data and does not contain metadata.
- **metadataOnly**
  Indicates that the disk contains metadata and does not contain data.
- **descOnly**
  Indicates that the disk contains no data and no metadata. Such a disk is used solely to keep a copy of the file system descriptor (FSDesc), and can be used as a third failure group in disaster recovery configurations.
- **FailureGroup**
  Is a number you can use to specifies in the disks which are inside the same physical enclosure
- **DesiredName**
  Is an user defined alias for the NSD to be created. This name must not already be used by another GPFS disk, and it must NOT start with the reserved string “gpfs”. Example 3-19 shows a sample GPFS disk descriptor file.

Example 3-19 Sample GPFS disk descriptor file

```
/dev/sde:::dataOnly:1:data1
/dev/sdf:::metadataOnly:1:metadata1
```

As there are no NSD servers specified, we can conclude that the GPFS direct attached model is used for this cluster. This means the disks will be directly attached to all nodes in the cluster.

Both disks belong to failure group one (1). But the disk usage is different: `/dev/sde` holds data only (and is named *data1*), and `/dev/sdf` holds metadata only (named *metadata1*).

This descriptor file will be used by several commands as input file. During this sequence of commands the file will also be changed (described later in more detail).

Tip: We recommend that you always specify disk failure groups. If you do not do so, GPFS will do it automatically, but this may lead to undesirable configurations as GPFS is not aware of the physical layout of the storage configuration. As a consequence, the failure of a single component may lead to unmounting affected file systems or even a cluster shutdown.

Example 3-20 shows all commands needed to set up a GPFS cluster. We also explain the commands step by step.

Example 3-20 Create and configure a GPFS cluster

```
mmcrcluster -n /tmp/gpfs.nodes -p gpfs41 -s gpfs43 -r /usr/bin/ssh -R
/usr/bin/scp -C GPFS-ITSO1
mmchconfig pagepool=512M,maxblocksize=1024K
```

Creating the cluster (mmcrcluster)

First, we create a cluster by using the **mmcrcluster** command. We specify the nodes and also designate at least one node as a quorum node. The nodes must be available for the command to be successful. We used a node descriptor file, named `gpfs.nodes` located in the /tmp directory (see Example 3-18 on page 131).

We decided to use node based quorum, therefore we assign three nodes as quorum nodes. The fourth node acts as GPFS client (default).

Important: Choose the quorum nodes carefully. An odd number of quorum nodes is recommended. For details, check *General Parallel File System (GPFS) for Clusters: Concepts, Planning, and Installation*, GA22-7968.

With the option **-p** and **-s** we specify the cluster configuration data servers (primary and secondary), in our case gpfs41 and gpfs43.

We also specify **/usr/bin/ssh** and **/usr/bin/scp** for remote command execution and remote file copy (**-r** and **-R** options).

Finally we defined the name of the cluster using the -C option, in our case “GPFS-ITSO1”.

Changing the cluster configuration (mmchconfig)

With the **mmchconfig** command, we change the cluster configuration parameters. While not limiting this command to dedicated nodes, the specified parameters will influence all nodes within the cluster; for example:

```
mmchconfig pagepool=512M,maxblocksize=1024K
```

The *pagepool* parameter changes the cache size on all GPFS nodes to 512 MB (pagepool is used for I/O operations and data caching). As we plan to use large block sizes to decrease the number of IO requests to the disk system (and implicitly increase the overall throughput), we also maximum allowed block size for the file systems to 1MB (1024kB).

Tip: You can also specify a config file while using the **mmcrcluster** command with the **-c** option. See the mmfs.cfg.sample file located in /usr/lpp/mmfs/samples/ for further details.

3.4.2 Start GPFS and verify all nodes join the cluster

As the cluster is now configured in principle we need to bring the GPFS daemons (**mmfsd**) up on all nodes. This achieved by using the **mmstartup -a** command. This is a first checkpoint if gpfs works. You can easily check the status of the nodes of the gpfs cluster by using **mmgetstate -a** command. Example 3-21 shows the output. All nodes should show to be active. Otherwise you need to investigate what causes trouble.

Example 3-21 Status of the nodes of the gpfs cluster (output of mmgetstate command)

```
p5n1lp1:~ # mmstartup -a
p5n1lp1:~ # mmgetstate -a

Node number  Node name       GPFS state
-----------------------------------------
       1      gpfs41           active
       2      gpfs42           active
       3      gpfs43           active
       4      gpfs44           active
```

3.4.3 Create NSDs

Now we approach the disk level and reuse the already created LUNs coming from our DS4500 storage. As GPFS needs an unique identifier for the disks to be able to access from any node within the cluster the same disk, the **mmcrnsd** command labels the specified disks (Example 3-22):

Example 3-22 Creating the NSDs

```
p5n1lp1:~ # mmcrnsd -F /tmp/gpfs.disk -v no
.....
p5n1lp1:~ # mmlsnsd
```

We provide the information about the disks to use and their usage in the disk descriptor file (/tmp/gpfs.disk - see Example 3-19 on page 132).

The **mmcrnsd** command changes the specified input file as you seen in Example 3-23. This file will be used again as input file for creating the file system.

Example 3-23 sample gpfs disk descriptor file after the use of the **mmcrnsd** *command*

```
# /dev/sde:::dataOnly:1:data1
data1:::dataOnly:1
# /dev/sdf:::metadataOnly:1:metadata1
metadata1:::metadataOnly:1
```

Attention: You can later change the configuration with **mmchnsd** command, i.e., from a direct attached configuration to a mixed environment.

3.4.4 Create the file system

In this step we create the file system using the **mmcrfs** command, shown in Example 3-24:

Example 3-24 Creating the file system

```
p5n1lp1:~# mmcrfs /gpfs gpfsdev1 -F /tmp/gpfs.disk -B 1024K -m 1 -M 2 -r 1 -R 2
```

The mount point for our file system is `/gpfs` and the associated device name is "*gpfsdev1*". The previous disk descriptor file (`/tmp/gpfs.disk` - as modified by the **mmcrnsd** command) is used as input to specify which disks will be used by the file system. We have chosen a block size of 1MB. The last four parameters configure the GPFS replication. While the uppercase letters -M and -R specifying the maximum numbers of replicas for data (-R) and metadata (-M), the lowercase letters define how many replicas actually to use for data (-r) and metadata (-m).

Attention: We strongly recommend to enable GPFS replication at file system creation time, even if you do not use it. You may need this later, and the only way to configure it is to delete the file system and create a new one where replication is enabled.

3.4.5 Mount the file system

Finally mount the file system by using **mount /gpfs** (where /gpfs is the mountpoint of our filesystem). The output of the mount command is shown in Example 3-25 on page 136.

Example 3-25 Output of mount command (cross-check if gpfs is mounted)

```
p5n1lp1:~ # mount /gpfs <----- on all nodes. you can use distributed shell

p5n1lp1:~ # mount
/dev/sdd3 on / type reiserfs (rw,acl,user_xattr)
proc on /proc type proc (rw)
tmpfs on /dev/shm type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/dvd on /media/dvd type subfs
(ro,nosuid,nodev,fs=cdfss,procuid,iocharset=utf8)
usbfs on /proc/bus/usb type usbfs (rw)
/dev/gpfsdev1 on /gpfs type gpfs (rw,mtime,dev=gpfsdev1,autostart)
```

3.4.6 Checking the cluster and file system configuration

By using the **mmlscluster** and **mmlsconfig** commands you can verify the cluster configuration, as shown in Example 3-26.

Example 3-26 GPFS cluster configuration (using mmlscluster/mmlsconfig commands)

```
p5n1lp1:~ # mmlscluster

GPFS cluster information
========================
  GPFS cluster name:         GPFS-ITSO1.gpfs41
  GPFS cluster id:           723500818519789615
  GPFS UID domain:           GPFS-ITSO1.gpfs41
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    gpfs41
  Secondary server:  gpfs43

Node number  Node name    IP address       Full node name    Remarks
-------------------------------------------------------------------------
       1      gpfs41       10.10.100.41     gpfs41            quorum node
       2      gpfs42       10.10.100.42     gpfs42            quorum node
       3      gpfs43       10.10.100.43     gpfs43            quorum node
       4      gpfs44       10.10.100.44     gpfs44

p5n1lp1:~ # mmlsconfig
Configuration data for cluster GPFS-ITSO1.gpfs41:
-------------------------------------------------
clusterName GPFS-ITSO1.gpfs41
clusterId 723500818519789615
```

```
clusterType lc
multinode yes
autoload no
useDiskLease yes
maxFeatureLevelAllowed 808
pagepool 512M
maxblocksize 1024K

File systems in cluster GPFS-ITSO1.gpfs41:
------------------------------------------
/dev/gpfsdev1
```

4

Advanced topics

This chapter presents advanced topics for GPFS installations and for administration in general environments. We also describe how to implement file sharing for clients that are not running GPFS, which is typical for a digital media environment.

The topics include:

- Linux multipathing
- SSH key distribution
- Configuring NFS on GPFS nodes
- Using NFS to share GPFS file systems

4.1 Linux multipathing

The term multipathing refers to a host accessing a storage (LUNs) in a SAN environment via multiple paths, the path being a route passing from the system through one fibre channel Host Bus Adapter (HBA) port, the SAN, and one storage controller port, up to the target LUN.

In a SAN environment a host can have multiple HBA installed for redundancy and/or load sharing, connected to the same SAN together with the storage subsystem. The storage subsystem itself usually has multiple Fibre Channel (FC) ports connected to the same SAN for redundancy and/or for load balancing. The storage subsystem provides LUNs for the attached systems, and as there are multiple paths from the host to each LUN, if there is no multipathing software installed on the host operating system, the same LUN will be “seen” multiple times, making difficult to manage the storage.

As Linux lacked a multipathing implementation for a long time, this has caused various implementations as temporary solutions. First there were vendor specific HBA driver multipathing solutions, then an RDAC driver was developed, but it has undergone several iterations before it was considered mature and safe to use. Meanwhile, the Linux kernel developers, together with some HBA hardware vendors, implemented a vendor-agnostic multipathing driver as part of the device mapper subsystem.

Currently there are three different multipathing implementations:

- Vendor specific HBA driver
- RDAC multipath driver
- Device mapper multipath driver

The following three figures (4-1, 4-2, and 4-3) present the structure of the HBA driver device access and management for a Linux environment. Figure 4-1 on page 141 shows the device driver without multipathing.

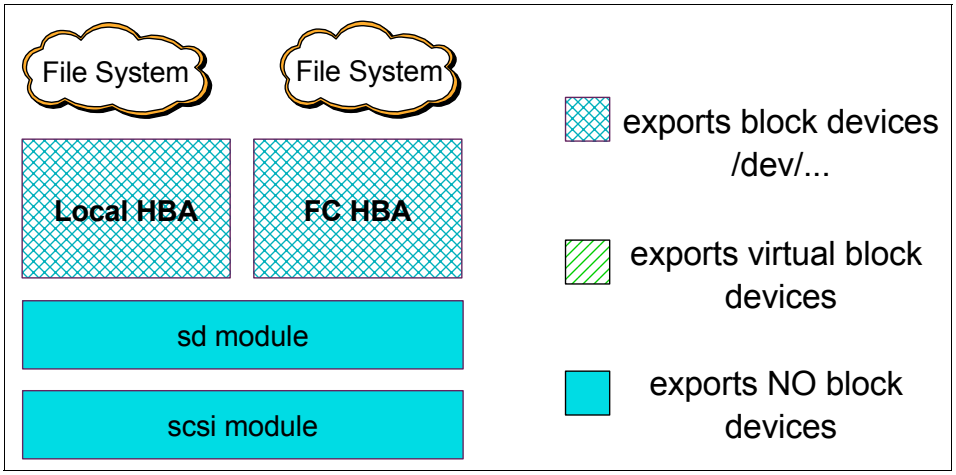

Figure 4-1 HBA modules without multipathing

The following Figure 4-2 presents the interaction between the HBA module(s) and the Linux (kernel) device mapper. The device mapper checks if the same target LUN is accessible via multiple paths and assigns a unique device entry (name) for each LUN.

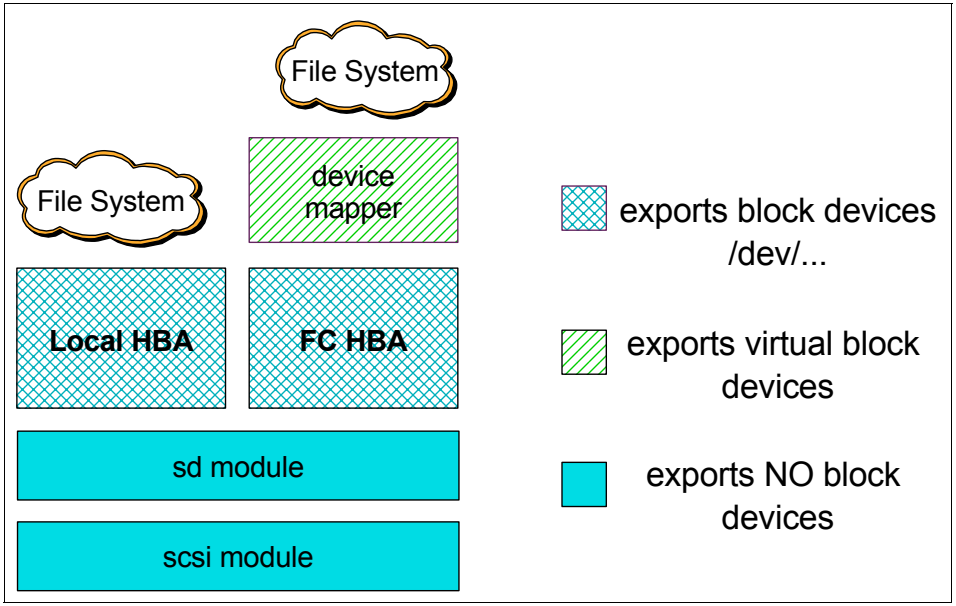

Figure 4-2 HBA modules with Linux device mapper

The Figure 4-3 shows the RDAC driver structure. RDAC (Redundant Disk Array Controller) is different from the device mapper in the sense that is has been tailored for specific (IBM) storage subsystems, and works closely with the storage subsystems to achieve both high availability and load sharing.

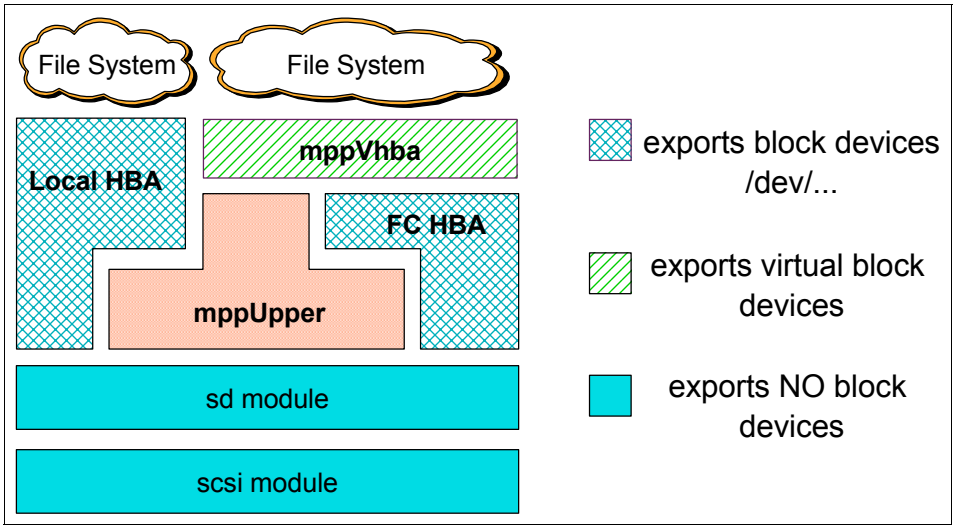

Figure 4-3 HBA modules with RDAC

The vendor specific device drivers work in certain operating system/storage subsystems combinations, but not all vendors provide this type of device drivers for their HBA products.

4.1.1 Using a vendor specific HBA multipath driver

Without any multipathing functionality, an HBA driver recognizes each path to the same LUN as a separate block device to the operating system. This means that if you have four distinct paths for the same storage array, each LUN provided by that storage array will be seen by the operating system (Linux) four times (as a different device - entry in /dev).

This is difficult to manage and undesired, as it allows concurrent access from the same node (by different applications) to the same physical device using different block devices, which could be potentially dangerous. One way to avoid this issue is to use *LUN masking* at device driver level (provided by some HBA vendors), which allows to mask all but one path for each physical storage LUN (in an external array). This implementation does not provide high availability or load sharing for storage devices.

Some vendors also provide a failover driver, that adds logic on top of the default functionality and merges these paths into a single block device. The driver allows the selection of the path to be used, and also allows a mechanism to select priorities for each path in case the path in use fails. You can control which path is normally used and the failover order, and whether to automatically failback in case the initial path is restored.

For this implementation you have to be very careful when selecting paths, since not all storage subsystems allow concurrent access to the same disk/array from different controllers (DS4500 for example). If not properly configured, in case you have multiple systems accessing the same disk/array, and one of them is performing a failover, this could result in bringing down the entire storage due to *LUN trashing*.

Important: LUN trashing occurs when the storage subsystem is transferring a disk/array from one controller to the other continuously. The result of LUN trashing is that the I/O operations are delayed (and possibly halted), and the LUNs become inaccessible.

AVT (Automatic Volume Transfer) is the mechanism that is causing a disk/array to be transferred from one controller to the other if requested by the system.

4.1.2 Using RDAC multipath driver

The RDAC driver is a solution to LUN trashing and simplifies the configuration of vendor specific HBA multipath driver.

The RDAC multipath driver consists of two parts, one part (**mppUpper**) is loaded before any HBA is loaded and prevents the HBA to advertise the LUNs to the system, while the other part (**mppVhba**) is a virtual HBA on top of the real HBA which is responsible for bundling the different paths to the same LUN to a single (multipath) device.

The RDAC driver can be downloaded from:

`http://www.ibm.com/servers/storage/support/disk/ds4500/stormgr1.html`

You need to download the latest package (check also the supported version against the GPFS F.A.Q. list) and transfer the package (.tar.gz archive) to each system that needs the driver installed.

For reference see 3.2.6, “ITSO lab environment overview” on page 102, and Figure 3-12 on page 103. Our cluster configuration consists of pSeries servers running SUSE SLES9 Service Pack 1, each with two HBAs (Emulex), connected via two 2109-F32 to a DS4500 with four host-side minihubs.

Note: We recommend that you save a copy of the existing initial ramdisk before installing the RDAC driver. This backup copy together with a separate stanza in /etc/lilo.conf are needed in case the system does not boot after installing the RDAC device driver.

The installation of the RDAC driver is shown in the following Example 4-1:

Example 4-1 Installing the RDAC driver

```
node01:~ # tar -xvzf rdac_LINUX_09.01.B5.02.tar.gz -C /usr/src/

node01:~ # make -C /usr/src/linuxrdac-09.01.B5.02/ clean uninstall all install
make: Entering directory `/usr/src/linuxrdac-09.01.B5.02'
make V=1 -C/lib/modules/2.6.5-7.139-pseries64/build
M=/usr/src/linuxrdac-09.01.B5.02 clean
make[1]: Entering directory `/usr/src/linux-2.6.5-7.139-obj/ppc64/pseries64'
make -C ../../../linux-2.6.5-7.139 O=../linux-2.6.5-7.139-obj/ppc64/pseries64
clean
...
Creating new MPP initrd image...
Module list:    scsi_mod sd_mod sg mppUpper sym53c8xx mppVhba reiserfs
Kernel version: 2.6.5-7.139-pseries64 (ppc64)
Kernel image:   /boot/vmlinux
Initrd image:   /boot/mpp-2.6.5-7.139-pseries64.img
mkdir: cannot create directory `/var/tmp/mkinitramfs.i10726/mnt/etc': File
exists
Shared libs:    lib/ld-2.3.3.so lib/libc.so.6 lib/libselinux.so.1
Modules:        kernel/drivers/scsi/scsi_mod.ko kernel/drivers/scsi/sd_mod.ko
kernel/drivers/scsi/sg.ko k
ernel/drivers/scsi/mppUpper.ko kernel/drivers/scsi/sym53c8xx_2/sym53c8xx.ko
kernel/drivers/scsi/mppVhba.k
o
7610 blocks
...
MPP driver package has been sucessfully installed on your system.
make: Leaving directory `/usr/src/linuxrdac-09.01.B5.02'
```

For details see the README file that comes with the package. Next, add the following section to /etc/lilo.conf (Example 4-2):

Example 4-2 Example RDAC section to be added to /etc/lilo.conf

```
image = /boot/vmlinux
    label = RDAC
    root = /dev/sda3
    initrd = /boot/mpp-2.6.5-7.139-pseries64.img
```

```
    append = "selinux=0 elevator=cfq
panic=5"
```

Substitute */dev/sda3* by the root partition you are using. To boot with the RDAC driver enabled, you have to change the default option to RDAC.

Note: It is important to **add** another section to the /etc/lilo.conf file, as this will allow you to revert to a previous configuration in case you run into boot problems when loading the new RDAC driver.

Even when using yaboot, you have to add the previously mentioned section to /etc/lilo.conf since SLES9 includes **`lilo`** as a script that converts /etc/lilo.conf into /etc/yaboot.conf (overwriting any changes you make to /etc/yaboot.conf). Also, as the last step, you should run the **`lilo`** command.

To load the RDAC driver, reboot and, if you have not selected the “RDAC” label as default boot configuration, type *RDAC* (the label you have assigned for this configuration in /etc/lilo.conf - see Example 4-2 on page 144) at the **`yaboot`** command prompt.

Attention: The RDAC documentation instructs you to change yaboot.conf directly, however any changes to yaboot.conf will be lost if you run **`lilo`**. Therefore, we recommend to make the changes to /etc/lilo.conf and then run **`lilo`**.

Yaboot (like LILO and GRUB) are boot managers, they allow a system to boot different configurations: Linux kernels, Linux distributions or even other operating systems. Since our environment is PPC (POWER5™ based), a different boot loader (Yaboot) is required than for x86 architecture. More information about Yaboot can be found at:

`http://penguinppc.org/bootloaders/yaboot/`

Problem determination

We have experienced problems when using the RDAC multipath driver in combination with the internal SCSI disk drivers. These problems include failure to boot from internal disk, failure to load the RDAC driver due to missing symbols and even a kernel panic. To analyze the problem, here are a few things you can check:

- Make sure the `sd_mod` and `sg` modules are loaded first (and not the `sr_mod` module), otherwise `mppUpper` might fail to load.

- Make sure the `mppUpper` module is loaded before any other HBA driver (e.g., `ipr`, `mptscsih`, `ips`, `sym53c8xx`, `lpfcdd`, `qla2xxx`), otherwise kernel panic may occur.
- Make sure the local SCSI disk driver is loaded before any other HBA driver, otherwise you might have problems booting your system because the local disk is named differently
- Make sure the FC-attached HBA driver is loaded before the mppVhba module and that it actually is included in the initial ramdisk, otherwise you have to unload and reload the mppVhba module manually

So the correct order to load the modules is:

1. Generic SCSI modules: `scsi_mod`, `sd_mod` and `sg`
2. RDAC Upper Level module: `mppUpper`
3. Local HBA modules: e.g., `ipr`, `mptscsih`, `ips` or `sym53c8xx`
4. FC attached HBA modules: `lpfcdd` or `qla2xxx`
5. RDAC Virtual HBA module: `mppVhba`

There is an easy way to verify that the created initrd follows these rules. The following output in Example 4-3 shows a wrong loading sequence (`lpfcdd` is loaded before `ipr`):

Example 4-3 Finding the load order of SCSI modules in the initrd

```
node01:~ # zcat /boot/mpp-2.6.5-7.139-pseries64.img | strings | grep '^insmod
/lib'
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/scsi_mod.ko
$extra_scsi_params max_report_luns=256
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/sd_mod.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/sr_mod.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/mppUpper.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/lpfc/lpfcdd.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/base/firmware_class.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/ipr.ko
insmod /lib/modules/2.6.5-7.139-pseries64/kernel/drivers/scsi/mppVhba.ko
```

The output shown in Example 4-4 presents the correct module loading sequence:

Example 4-4 Finding the load order of SCSI modules in the initrd

```
node01:~ # zcat /boot/mpp-2.6.5-7.145-pseries64.img | strings | grep '^insmod
/lib'
```

```
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/scsi/scsi_mod.ko
$extra_scsi_params max_report_luns=256
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/scsi/sd_mod.ko
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/scsi/sg.ko
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/scsi/mppUpper.ko
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/base/firmware_class.ko
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/scsi/ipr.ko
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/scsi/lpfc/lpfcdd.ko
insmod /lib/modules/2.6.5-7.145-pseries64/kernel/drivers/scsi/mppVhba.ko
```

If you have trouble generating the `initrd` to obey these rules on SLES9, you can force it, using the following command (here `ipr` is the driver for the local SCSI disks and `lpfcdd` is the FC-attached storage):

```
mkinitrd -m "sd_mod sg mppUpper ipr lpfcdd mppVhba" -i "mpp-$(uname
-r).img" -k "vmlinux"
```

To check the results of loading the RDAC driver, use the **`lsscsi`** command output to understand what the resulting block devices are when using the RDAC driver (Example 4-5):

Example 4-5 Using lsscsi to relate block devices to the RDAC driver (VirtualDisk)

```
node01:~ # lsscsi
[0:0:8:0]    disk    IBM      HUS103014FL3800  RPQF  /dev/sda
[0:0:9:0]    disk    IBM      HUS103014FL3800  RPQF  /dev/sdb
[0:0:15:0]   enclosu IBM      HSBPD4E  PU3SCSI 0018  -
[2:0:1:0]    cd/dvd  IBM      DVRM00203        A151  /dev/sr0
[4:0:0:0]    disk    IBM      1742-900         0520  -
[4:0:0:1]    disk    IBM      1742-900         0520  -
[4:0:1:0]    disk    IBM      1742-900         0520  -
[4:0:1:1]    disk    IBM      1742-900         0520  -
[5:0:0:0]    disk    IBM      VirtualDisk      0520  /dev/sdc
[5:0:0:1]    disk    IBM      VirtualDisk      0520  /dev/sdd
```

4.1.3 Using device mapper multipath driver with GPFS

The Linux 2.6 kernel comes with the ability to do multipathing on top of existing detected LUNs. Even though device mapper multipathing is not (yet) supported by GPFS, it is the Linux vendor's recommended way of doing device-driver independent multipathing.

Attention: We have tested the device mapper driver in our environment. This does NOT imply official IBM support for this configuration. Always check the latest supported configurations at:

`http://www.ibm.com/servers/eserver/clusters/software/gpfs.html`

Click the GPFS F.A.Q. link.

Figure 4-4 shows a two node cluster configuration where each host has two HBAs connected (direct or via s SAN fabric) to a storage subsystem with two paths to every LUN. In this figure you can see that there are four distinct paths to every LUN provided by the storage subsystem.

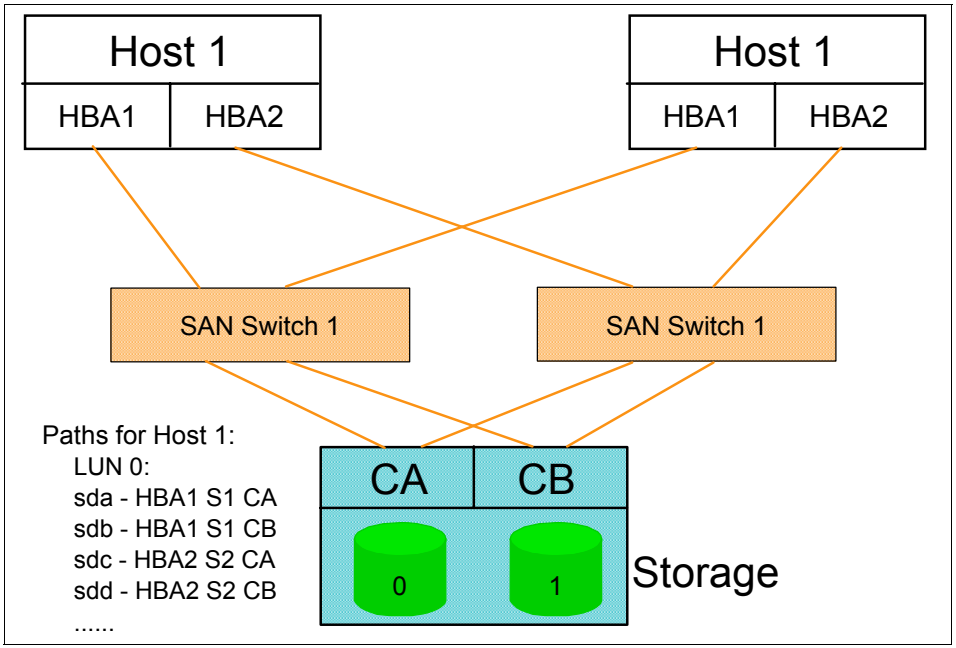

Figure 4-4 Cluster configuration with redundant storage paths

The devices `sda`, `sdb`, `sdc`, and `sdd` are in fact the same LUN (identified by a unique world wide number - WWN), seen four times via four distinct paths.

The Linux kernel (or HBA device driver) does not mask the different paths, so each LUN will appear several times as a SCSI disk on the system (/dev/sdX). The device mapper multipath driver is a kernel module within the device mapper subsystem. This device driver identifies the LUNs by their WWN and maps all /dev/sd* with the same WWN into a /dev/dm-* device.

The application (file system in this case) uses the /dev/dm-* devices instead of /dev/sd* ones (see Figure 4-5 on page 149).

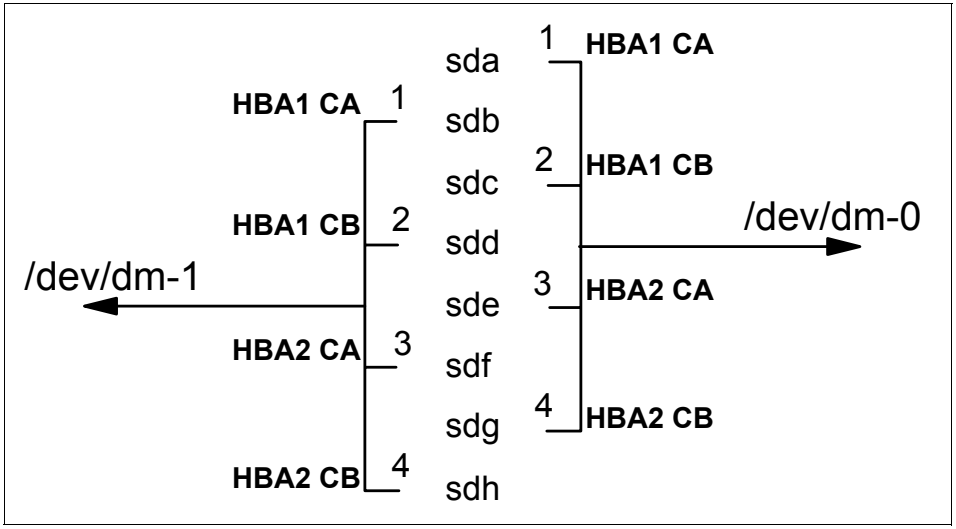

Figure 4-5 Linux device mapper example with two LUNs

To be able to use device mapper for multipathing, you need a kernel with device mapper support, the device mapper multipath module (`dm-multipath`) and the multipath tools package which contains the multipath commands, sample configuration files and driver documentation.

In addition to the device mapper module, there are two user space utilities that influence the device mapper multipath behavior. These two utilities are the **`multipath`** command and the **`multipathd`** daemon (see Figure 4-6 on page 150).

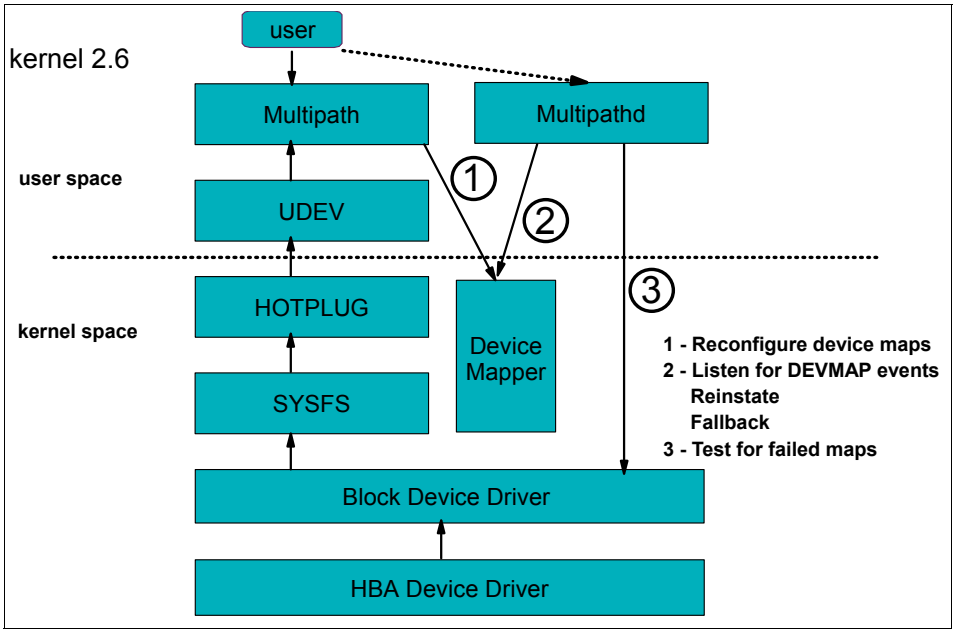

Figure 4-6 Linux device mapper structure

The `multipath` command

The **`multipath`** configuration tool is responsible for collecting all the different paths that address a LUN and creating the device map. A device map is the definition of the different paths (physical LUNs) and the path distribution policy for accessing the LUN. The **`multipath`** command is triggered by the hotplug subsystem whenever it detects a change (paths appearing or disappearing).

There are several available distribution (spreading) policies. Depending on the multipath version shipped with your Linux distribution and service pack, the policies list and names differ slightly.

- **Failover**: only a single path is used for I/O, the other one(s) is in standby
- **Multibus**: all the paths are used for I/O
- **group_by_serial**: a single path is used for I/O for a specific storage controller

Depending on the storage subsystem used, it is possible to access the same LUN using different storage controllers simultaneously without any restriction. In that case **multibus** is an option, if there is a restriction when moving to the LUN to another controller, then the **group_by_target** or **failover** policy is preferred.

If you run the **`multipath`** userspace tool, the output will looks similar to the one shown in Example 4-6 on page 151:

Example 4-6 Verbose multipath output

```
node01:~ # multipath -v2 -pfailover
#
# all paths :
#
3600a0b800012106e0000000a42875589 (9 0 0 0) sdc [ready ] (8:32) [1742-900 ]
3600a0b800012106e0000000c4287559b (9 0 0 1) sdd [ready ] (8:48) [1742-900 ]
3600a0b800012106e0000000a42875589 (9 0 1 0) sde [ready ] (8:64) [1742-900 ]
3600a0b800012106e0000000c4287559b (9 0 1 1) sdf [ready ] (8:80) [1742-900 ]
3600a0b800012106e0000000a42875589 (10 0 0 0) sdg [ready ] (8:96) [1742-900 ]
3600a0b800012106e0000000c4287559b (10 0 0 1) sdh [ready ] (8:112) [1742-900 ]
3600a0b800012106e0000000a42875589 (10 0 1 0) sdi [ready ] (8:128) [1742-900 ]
3600a0b800012106e0000000c4287559b (10 0 1 1) sdj [ready ] (8:144) [1742-900 ]
#
# all multipaths :
#
3600a0b800012106e0000000a42875589 [1742-900        ]
 \_(9 0 0 0) sdc [ready ] (8:32)
 \_(9 0 1 0) sde [ready ] (8:64)
 \_(10 0 0 0) sdg [ready ] (8:96)
 \_(10 0 1 0) sdi [ready ] (8:128)
3600a0b800012106e0000000c4287559b [1742-900        ]
 \_(9 0 0 1) sdd [ready ] (8:48)
 \_(9 0 1 1) sdf [ready ] (8:80)
 \_(10 0 0 1) sdh [ready ] (8:112)
 \_(10 0 1 1) sdj [ready ] (8:144)
#
# device maps :
#
reload:3600a0b800012106e0000000a42875589:0 569298944 multipath 1 round-robin 4
0 8:32 8:64 8:96 8:128
reload:3600a0b800012106e0000000c4287559b:0 142324736 multipath 1 round-robin 4
0 8:48 8:80 8:112 8:144
```

The different sections in the **multipath** command output are:

- The first section in the command output displays the different paths found by the HBA device driver in the format: unique LUN id, the SCSI target (host/bus/target/lun quartet), the device name (sdX), the path state (Ready/Failed), the block device major:minor and the storage subsystem model.
- The second section shows the different paths for each unique LUN id.
- The third section shows the resulting device maps created by the multipath device driver and the action send to the device mapper. (Create/Reload).

The order of the device maps indicates the different device mapper devices (/dev/dm-X) available (to be used by a file system). In our example we have two LUNs, mapped to /dev/dm-0 and /dev/dm-.

The **group_by_target** device map has the following syntax:

```
1___2________3____4_____5_____6_7__8____9______10____11_12__13___14__
0 27262976 multipath 2 round-robin 2 0 8:48 8:112 round-robin 2  0 8:80 8:144
```

- field 1-2: start and end offset
- field 3: multipath identifier
- field 4: number of priority groups
- field 5-9: first priority group
  - field 5: the scheduler to be used to spread the IO inside the priority group
  - field 6: number of paths in priority group
  - field 7: number of path parameters (usually 0)
  - field 8-9: the paths for this priority group
    - 8:48 is sdd
    - 8:112 is sdh
- field 10-14: second priority group

Depending on the spreading policy, this syntax will adopt a number of priority groups and classify the different paths to each priority group. The given example indicates that this LUN has 4 paths divided into 2 priority groups, using a round-robin scheduler (`dm_round_robin`) for each, effectively using only the 2 paths of the first priority group.

Here is an example of a **failover** policy:

```
0 27262976 multipath 4 round-robin 1 0 8:48 round-robin 1 0 8:80 round-robin 1
0 8:112 round-robin 1 0 8:144
```

And, an example of a **multibus** policy:

```
0 142324736 multipath 1 round-robin 4 0 8:48 8:80 8:112 8:144
```

The `multipathd` daemon

The **`multipathd`** daemon is responsible for checking the availability of each path and if necessary reinstate paths and perform a failback. It checks the available paths regularly (polling) and also waits for events from the kernel module in case the device maps change.

The **multipathd** daemon will run **multipath** command on startup to create or reload the necessary device maps. If you rely on **multipathd**, you need to make sure **multipathd** is started when you boot the system.

The /etc/multipath.conf file

You can influence the multipath user space configuration tool behavior by creating the /etc/multipath.conf config file. This allows you to specify a spreading policy by storage subsystem vendor and model, and also allows you to name the aliases for each device map (see Example 4-7).

Example 4-7 Example /etc/multipath.conf file

```
defaults {
   multipath_tool  "/sbin/multipath -v 0 -S"
   udev_dir        /dev
   polling_interval 10
   default_selector        round-robin
   default_selector_args   0
   default_path_grouping_policy    failover
   default_getuid_callout  "/sbin/scsi_id -g -u -s"
   default_prio_callout    "/bin/false"
}

devnode_blacklist {
   devnode fd
   devnode hd
   devnode dm
   devnode sr
   devnode scd
   devnode st
   devnode ram
   devnode raw
   devnode loop
   devnode sda
   devnode sdb
}

multipaths {
   multipath {
      wwid    3600a0b800012106e0000000a42875589
      alias   data
   }
   multipath {
      wwid    3600a0b800012106e0000000c4287559b
      alias   metadata
   }
}
```

```
devices {
   device {
      vendor                "IBM     "
      product               "1742-900        "
      path_grouping_policy  group_by_serial
      path_checker          tur
   }
}
```

When using this configuration file, the **multipath** user space config tool will display in addition the alias for each multipath and it will filter out the local (/dev/sda and /dev/sdb) disks as possible targets.

To check the multipath configuration you can use the **lsscsi** command, as shown in Example 4-8:

Example 4-8 Block devices to HBA and storage controller relationship

```
node01:~ # lsscsi
[0:0:8:0]    disk    IBM      HUS103014FL3800  RPQF  /dev/sda
[0:0:9:0]    disk    IBM      HUS103014FL3800  RPQF  /dev/sdb
[0:0:15:0]   enclosu IBM      HSBPD4E  PU3SCSI 0018  -
[2:0:1:0]    cd/dvd  IBM      DVRM00203        A151  /dev/sr0
[9:0:0:0]    disk    IBM      1742-900         0520  /dev/sdc
[9:0:0:1]    disk    IBM      1742-900         0520  /dev/sdd
[9:0:1:0]    disk    IBM      1742-900         0520  /dev/sde
[9:0:1:1]    disk    IBM      1742-900         0520  /dev/sdf
[10:0:0:0]    disk    IBM      1742-900         0520  /dev/sdg
[10:0:0:1]    disk    IBM      1742-900         0520  /dev/sdh
[10:0:1:0]    disk    IBM      1742-900         0520  /dev/sdi
[10:0:1:1]    disk    IBM      1742-900         0520  /dev/sdj
```

In Example 4-8 `sdc`, `sde`, `sdg` and `sdi` are the same physical LUN, using different paths (different HBA and storage controller combination). Since the device mapper block devices are not SCSI LUNs, they do not show up in the output.

Making GPFS device mapper aware

GPFS has an internal (“white”) list of device names can be used. This list includes common block device names like /dev/sdX or /dev/hdX, but /dev/dm-X device names are not included since these are not officially supported by IBM.

You can change this by creating the file */var/mmfs/etc/nsddevices*. Since this files is run like a shell script, we have to introduce the necessary devices instead of

listing them. The following Example 4-9 shows how to configure (up to) four (4) device mapper devices, if you need more you can just add them to the list in the /var/mmfs/etc/nsddevices file:

Example 4-9 Changing device probing in GPFS using /var/mmfs/etc/nsddevices

```
echo dm-0 device-mapper
echo dm-1 device-mapper
echo dm-2 device-mapper
echo dm-3 device-mapper
```

Since GPFS adds a unique identifier to each disk (NSD id) each node will be able to find the correct device mapper block device for each NSD even when the order of the device mapper devices is different.

Attention: If you want to mix device mapper with other SCSI devices you have to make sure that you do not overlap the LUNs that are already part of the existing device mapper devices. In this case you can only include those devices (dm-X and sdX) that you plan to use, since GPFS might otherwise find a SCSI LUN instead of the device mapper device during scanning for NSD ids.

Device mapper multipathing limitations

- The system requires multiple LUNs for each device mapper device, for example if you have four paths, four LUNs will be used. Since the kernel/HBA driver usually is limited to 256 LUNs, this will reduce the number of multipath LUNs to only 64. Using an RDAC driver, or using a vendor specific (Qlogic, for example) failover driver with LUN masking will effectively use a single device per physical (storage provided) LUN, therefore allowing up to 256 LUNs.
- As the user only has only limited control over the paths the system is using, this limits the capability of creating and testing complex (multiple storage subsystems) environments.

More information about device mapper is at:

http://sources.redhat.com/dm/

More information about device mapper multipathing is at:

http://christophe.varoqui.free.fr/

4.2 SSH key distribution

GPFS requires remote command execution from one node to another (as root user) without using a password (non interactive). As we do not recommend the

use of classic remote shell (**rsh**), we describe in this section how to configure secure shell. There are several ways of achieving this goal, but the basic principle is to use the public key of the remote identity (username@host) as credentials verification for remote command execution (see Figure 4-7).

To achieve this, you need to add the user's public key from the source host to the authorized_keys file of the identity (user@hostname) on the target host. In addition to this, **ssh** will also verify of the system key matches a previously stored system key in known_hosts, in order to prevent a man-in-the-middle attack (someone spoofing the target system to trick you).

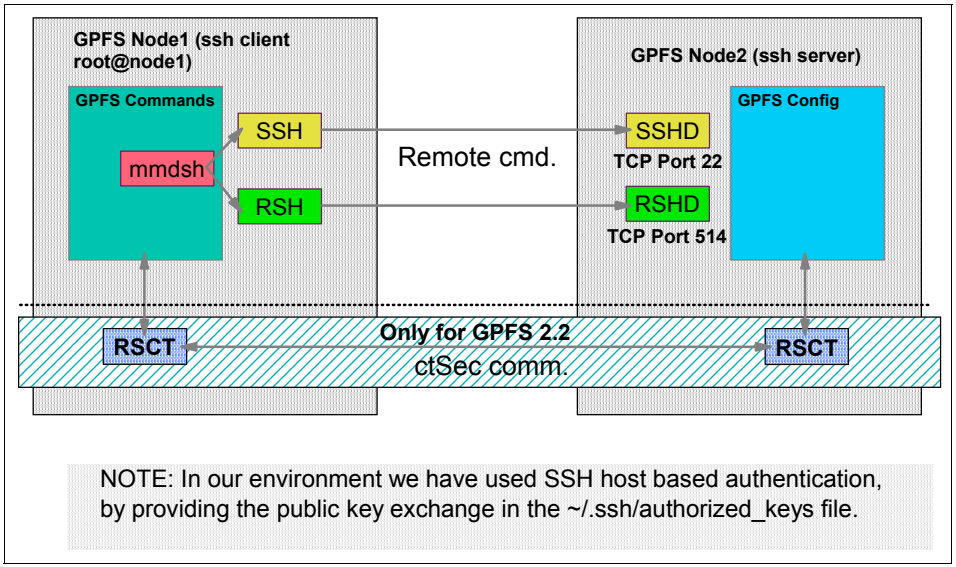

Figure 4-7 Security communication in a GPFS cluster

When using default **ssh** configuration, there is set of keys for the root user on each node. For example, in a ten-node cluster, this results in the necessity for the systems administrator to manage 20 different keys (10 for the daemons - sshd, and 10 for the clients) on each system, making it difficult to replace or to add a node to the cluster.

To simplify the generation and management of these keys, we have used a single set of **ssh** client keys for the root user, and distributed these to each node. This simplifies initial node setup and further node operations.

Prepare the system for SSH on one designated node

Designate one node for creating the **ssh** key set and managing the different files that need to be keep up to date and in sync on all nodes. We use *node01* for this role.

As user root, create an ssh key on node01, without a passphrase:

```
ssh-keygen -t dsa -N ""
```

This will create two keys in /root/.ssh/: id_dsa (the private key) and id_dsa.pub (the public key).

Preparing passwordless logon

Passwordless logon using ssh normally requires that the public key identity (username@your_source_host) is stored in the authorized_keys of the target login identity on the target host (in our case root@nodeX). To simplify this, we consider node1 also as the target host and install the public key directly in our own authorized_keys file:

```
cat /root/.ssh/id_dsa.pub >>/root/.ssh/authorized_keys
```

Populating the known_hosts file

Beside the ~/.ssh/authorized_keys file, a ~/.ssh/known_hosts file containing the public key of the ssh daemons running on all nodes in the cluster must be created. A simple way of collecting these public keys is to login to all nodes from the designated node (including the local node). This populates the local (in this case, on node01) ~/.ssh/known_host file.

Note: If you have multiple interfaces (IP addresses) on each node, you have to logon from your designated administrative node (in our example, node01) to all nodes using all available interfaces. Failing to do so may result in GPFS failing distributed operations.

Distributing the ssh authentication files and testing

Once your known_hosts file is populated, you have to distribute the contents of ~/.ssh/ directory to all the other nodes:

```
scp -rp /root/.ssh root@node02:
scp -rp /root/.ssh root@node03:
...
```

Once these files are copied, you should be able to use **ssh**, **scp** or **shmux** to any other node without providing a password. To verify, we recommend to try to logon from every node to all nodes (without a password).

4.3 Samba server

Samba in an Open Source CIFS (Common Internet File System) implementation. CIFS is also known as SMB (Shared Message Block) or as the

Windows network protocol that allows file and printer sharing in a Windows network.

4.4 NFS

The Network File System (NFS) is choice for file sharing in the UNIX world (as is Samba the in Windows environments). NFS allows computers to access files over an IP network as though they were on the local disk (file system).

NFS has evolved from a insecure stateless protocol to a secure stateful protocol. The latest version, NFS version 4, introduces byte range locking, share reservation, file delegation (similar to oplocks in Samba), and single roundtrip commands.

NFS relies on remote procedure calls (RPCs) for communication between clients and servers. NFS is UNIX-derived and therefore implements a UNIX-style set of file permission, limited a combination to the three types of ownership (user/group/other) and three types of file permission (read/write/execute).

NFS file server can apply various options (e.g., restricting all access to read-only) when exporting, and an NFS client has similar options when mounting from an NFS file server.

NFS export can be specified by NFS clients, netgroups (lists of clients), or even subnets.

The NFS implementation in Linux is kernel based, which improves performance as the kernel does not have to copy data from/to user space.

Originally developed by Sun™ Corporation (http://www.sun.com), it has been adopted as the de-facto standard file sharing protocol/application. There is a lot of information about NFS for Linux on the Internet. The following links provide a good starting point:

http://nfs.sourceforge.net/
http://nfs.sourceforge.net/nfs-howto/

4.4.1 Installing NFS server on Linux

On both SLES and RHEL, the NFS server is part of the *nfs-utils* package. To install it:

On SLES use:

```
yast -i nfs-utils
```

On RHEL, you have to issue:

```
up2date -i nfs-utils
```

4.4.2 Configuring NFS

Both distributions come with front-ends to configure the NFS server. SLES uses a *yast2* module (*yast2-nfs-server*) and on Red Hat you have the *system-config-nfs* package.

We do not describe these graphical frontends in this section. However, both tools provide you access to the list of exported file systems. This is stored in /etc/exports and contains a lists of all exported file systems, each on a separate h line, including the access list and options.

An example of such an exported file system is:

```
/gpfs           *(rw,async,insecure,no_wdelay,root_squash)
/dev/shm        *(rw,async,insecure,no_wdelay,root_squash)
```

Tip: We use /dev/shm here as an example. The /dev/shm directory is a dynamic ramdisk which allows you to run system performance tests that are not influenced by the actual disk subsystem. The size of this ramdisk however is limited to half of the system's physical RAM and only the actual used space is claimed from the system.

The options can also be set on a per-client basis instead of on a per-filesystem basis. To do this, you can specify it per client like this:

```
/path/foo    client1(option1,option2)     client2(option1,option2)
```

The client part can be an IP address (1.2.3.4), a resolvable hostname (client1.my.company), a network address (1.2.3.0/24 or 1.2.3.) or even part of a resolvable domain name (*.my.company).

4.4.3 Starting NFS

You can start the NFS server on SLES by running:

```
/etc/init.d/nfslock start
/etc/init.d/nfsserver start
```

On Red Hat you need to start:

```
/etc/init.d/nfslock start
/etc/init.d/nfs start
```

If everything loaded correctly, your kernel should have inserted the *exportfs* and *nfsd* kernel modules. Since the NFS server is kernel based, it writes information about requests directly to syslog, so you can find logging information in /var/log/messages.

To have the NFS server start automatically when you boot the system, use the following commands:

```
chkconfig nfslock on
chkconfig nfsserver on
```

On Red Hat you need to start:

```
chkconfig nfslock start
chkconfig nfs start
```

Every time you add new exported filesystems or change options in /etc/exports, you need to make the NFS server aware of these changes. You can do this with:

```
exports -r
```

4.4.4 NFS in a digital media environment

For NFS you have three levels of controlling the behavior of NFS connections:

- You can influence NFS globally by starting the NFS daemon with certain options.
- You can influence the behavior on a per-filesystem, per-ACL basis by changing options in /etc/exports.
- The client can influence the behavior of the connection providing certain mount command options.

In the following paragraphs, we present where each of these changes applies.

Protocol to use: UDP or TCP

Starting with NFS version 3 (and some custom implementations of NFS V2) you can use TCP as a transport instead of UDP. TCP has an important advantage over UDP, as it is connection oriented and guarantees delivery of the packages, without the need to retransmit the complete RPC request. Also, because of flow control of the underlying network level, TCP handles the network speed differently (and better) than UDP.

The disadvantage of using TCP is the protocol overhead, and also if a server crashes, this can cause the client to hang. A server crash will cause processes that are writing to an NFS share to hang in “D” state (uninterruptable sleep) until the NFS server is recovered.

Nevertheless, we recommend using TCP when using NFS. You can specify to use tcp or udp, by using it as a mount command option, like:

```
mount //node01/gpfs /gpfs -o tcp
```

Block size

The mount command options *rsize* and *wsize* specify the data block size that the NFS client is sending, receiving respectively, to/from the NFS server. If no **rsize** or *wsize* values are provided, a default value is used (which is most likely 4kB, but can vary from system to system). Starting with Linux 2.4 kernel it is possible to have the server provide the default block size.

The block size value depends on the specific combination of hardware, kernel and intermediate network devices. Therefore, we recommend that you carefully test and select an optimal value for your specific environment.

```
mount //node01/gpfs /gpfs -o rsize=65536,wsize=65536
```

Synchronous versus asynchronous

The default NFS communication behavior is asynchronous. This allows the server to reply to client requests before previous requests have been fully written (committed) to the disk. In this way, NFS is able to achieve much better performance than when using synchronous communication.

The disadvantage of the asynchronous option is that if a server reboots while still holding unwritten data or metadata in its caches, your data on disk might get corrupted. It's not possible for the client to detect this potential data corruption as the server does not informed the client if the data has actually been written to disk.

Number of NFS kernel threads

The default number of NFS kernel threads on SLES9 is four (4) per processor. For a dedicated NFS server, you might want to increase this value to 8 threads per processor, depending on the number of clients using each system.

You can set this value in `/etc/sysconfig/nfs`, the correct syntax for this on SLES is:

```
USE_KERNEL_NFSD_NUMBER="8"
```

And, on RHEL:

```
RPCNFSDCOUNT="8"
```

Forcing the NFS version

When mounting an NFS filesystem, you can specify what NFS protocol version the client has to use. This allows to test/use different versions depending on what functionality you require. You can do this with the **nfsver** mount command option.

```
mount //node01/gpfs /gpfs -o nfsver=3
```

Hard or soft mount

NFS allows to either hard mount or soft mount an NFS filesystem, the difference is that when hard mounting NFS it cannot be interrupted when the server becomes unavailable. Depending on the environment, this can be either an advantage or a disadvantage.

In case your server becomes unavailable, and applications use a hard mounted NFS file system, the applications will wait for the NFS server to come back (as long as it takes). If the server is not coming back, the applications cannot be interrupted at all and even may require a system reboot to bring them back to a consistent state.

When an NFS filesystem is soft mounted, a specified timeout will cause the kernel to drop the mount if the NFS server is not responding. This might be useful if the server sometimes stops responding or will be rebooted, but it may also cause applications to fail in case NFS server comes back after a while.

We recommend hard mounting NFS file systems and make sure the NFS server is always available and responding.

User mapping

NFS expects (or even requires) that user IDs and group IDs are the same on all the systems that share the exported file systems, which may also be valid for the application using that file system. NFS bases its access control on this assumption, however this is not always the case and may not even be desirable.

For security reasons often is not desired nor recommended that a root user on the client machine has root access on the files on an exported NFS file system. This is why NFS by default maps uid 0 (root uid) to a different uid, usually an anonymous or nobody id, or to a specified uid/gid. This is what often is referred to as *root squashing*. You can disable root squashing by specifying **no_root_squash** on a per file system basis.

You can map all user request to the anonymous uid with the **all_squash** option.

A Windows NFS implementation usually offers a different way of mapping user names to user IDs, and you can specify this policy (or a fixed user) using the Windows GUI.

Additional information NFS

For Linux NFS performance tuning you can check the following documents:

- The Performance of a Linux NFS Implementation: at:

  http://www.wpi.edu/Pubs/ETD/Available/etd-0523102-121726/unrestricted/boumenot.pdf

- Linux NFS Client Write Performance, at:

  http://www.citi.umich.edu/techreports/reports/citi-tr-01-12.pdf

For AIX NFS performance tuning refer to “AIX Performance Management Guide”, SC23-4905, or online at:

http://publib.boulder.ibm.com/infocenter/pseries/index.jsp?topic=/com.ibm.aix.doc/aixbman/prftungd/prftungd.htm

4.4.5 Integrating NFS and GPFS

This section describes some aspects to be considered when NFS exporting a GPFS file system.

Creating a runtime dependency between NFS and GPFS

In certain situations GPFS might not be available (not running), but NFS might still be active. In this case users, not aware of this downtime, might try to use the NFS shares and end up inside the (unmounted) mountpoint, effectively using the local disk instead of the shared storage.

This can cause unexpected application behavior or users might be unknowingly adding new files to this empty directory on their local disk. This situation has to be prevented, and the easiest way to do that is to disable automatic NFS activation at system startup, and let GPFS take care of starting and stopping the NFS services.

GPFS allows to use (plug in) your own commands during start-up and shut-down. The files which allow these tasks are `/var/mmfs/etc/mmfsup.scr` and `/var/mmfs/etc/mmfsdown.scr`

NFS has a special per file system option, called **`mountpoint`** or **`mp`** that will ensure that the filesystem will only be exported if the provided path is actually a mount point. This helps to prevent exporting a non mounted GPFS mountpoint, but will not provide the same protection in case GPFS is stopped (shutdown), while the file system is already exported by NFS. However, we recommend to use this parameter, as it is preferred to not have the wrong filesystem exported.

First prevent the NFS server from starting at boot time (before GPFS), by doing on SLES:

```
chkconfig nfsserver stop
```

And on RHEL

```
chkconfig nfs stop
```

To let GPFS start the NFS server on startup, you can append the following line to (the end of) /var/mmfs/etc/mmfsup.scr for SLES:

```
/etc/init.d/nfsserver start
```

And for RHEL:

```
/etc/init.d/nfs start
```

To let GPFS shutdown the NFS server when it is down, you can add the following line to the section that has the following comment in the file /var/mmfs/etc/mmfsdown.scr, on SLES:

```
# Do any commands here to clean up local state
# when the mmfsd daemon is shutting down.
/etc/init.d/nfsserver stop
```

On RHEL:

```
# Do any commands here to clean up local state
# when the mmfsd daemon is shutting down.
/etc/init.d/nfs stop
```

Attention: There are two places in the mmfsdown.scr file that have an almost identical description, use the one that clearly states the *shutdown* phase, and not the *restart* phase.

NFS file system identification

NFS has a per file system file system ID option (*fsid*) used to identify NFS file systems across multiple systems. This is specially useful when using the same GPFS file system exported from multiple nodes. Usually the *fsid* defaults to a number derived from the major and minor device number of the underlying block device.

Important: As a GPFS file system has the same major/minor number on all cluster nodes, it is **mandatory** to specify the same *fsid* when NFS exporting a GPFS file system from multiple nodes.

The *fsid* value is a unique 32bit number and is useful in an NFS failover scenario, to ensure that both servers of the failover pair use the same NFS file handles for the shared filesystem, avoiding stale file handles after a failover.

In NFS V4 a value of 0 (zero) has a special meaning. NFS V4 has the capability of defining a root of the overall exported filesystem, and exporting a file system with a *fsid* of 0 (zero) will be used as this root.

Conclusions

Given the previously mentioned considerations, we recommend similar options for your digital media environment. For the block sizes, test and check that performance is actually improving before using any set of values in a production environment.

For the /etc/exports file:

```
/gpfs        client01(rw,root_squash,async,secure,mountpoint,fsid=1234)
```

And for the client mounting the NFS filesystem:

```
mount //node01/gpfs /gpfs -o rsize=65536,wsize=65536,tcp,hard
```

4.4.6 Preparing windows clients

One of the aspects you have to consider for Windows clients is the use of mixed case letters (for files and directories names). Check with the actual client application before deciding a policy to use. Also follow this minimal set of guidelines to configure Windows parameters:

- Check TCP/IP settings in the Windows registry (see Example 4-10)
- Disable indexing for the GPFS file system on all (windows) clients attached to the system. Another option is to completely disable the (Windows) indexing service.
- Consider how (Windows) virus scans mail impact the overall system. It is not a good idea to run virus scans from all (Windows) clients on the GPFS file system as this dramatically decreases performance.

Example 4-10 Windows registry settings for Samba/NFS clients

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters]
"TcpWindowSize"=dword:000ffff0
"Tcp1323Opts"=dword:00000003
"GlobalMaxTcpWindowsize"=dword:000ffff0
```

4.5 Additional GPFS management tasks

This section provides additional information about installing GPFS in a Linux cluster using slightly different tools and hardware than the procedure presented in 3.4, “GPFS installation and configuration” on page 126.

Configuring shmux (a distributed remote execution tool)

First we create a file that containing the node (network) names to use with **shmux**:

```
mkdir /etc/gpfs
echo -e "node01\nnode02\nnode03" >/etc/gpfs/shmux
echo "export SHMX=\"/etc/gpfs/shmux\"" >/etc/profile.d/shmux.sh
```

Then we create a ~/.bashrc file containing the environment variables to be loaded for the root user (see Example 4-11):

Example 4-11 Setting the environment variables in ~/.bashrc

```
cat <<EOF >/root/.bashrc
   #!/bin/bash
   export SHMX="/etc/gpfs/shmux"
   export PATH="$PATH:/usr/lpp/mmfs/bin"
   export GPFS_rshPath="/usr/bin/ssh"
   export GPFS_rcpPath="/usr/bin/scp"
   export PROMPT_COMMAND='echo -ne
"\033]0;${USER}@${HOSTNAME%%.*}:${PWD/#$HOME/~}\007"'
EOF
chmod u+x /root/.bashrc
```

You can make this change effective for the current shell using:

```
source /root/.bashrc
```

Prepare the system for ssh on one node

Create an SSH key on one node, without a passphrase

```
ssh-keygen -t dsa -f ~/.ssh/id_dsa -N ““
```

Install the public key in the authorized_keys file on that node

```
cat /root/.ssh/id_dsa.pub >>/root/.ssh/authorized_keys
```

Test the logon to each node so that all keys are stored in known_hosts, do this for every interface that GPFS or yourself will be using

```
ssh node01
ssh node02
ssh node03
```

```
...
```

Copy the complete contents of the SSH directory to the other nodes

```
scp -rp /root/.ssh root@node01:
scp -rp /root/.ssh root@node02:
scp -rp /root/.ssh root@node03:
...
```

Or by using **shmux**:

```
shmux -c "scp -rp root@/root/.ssh/.??* /root/.ssh/" -<$SHMX
```

For details, see also 3.3.4, "Configuring secure shell" on page 122.

Installation of the Qlogic driver

In the example we have presented in Chapter 3, "Infrastructure configuration" on page 81 we have used Emulex HBA (the default choice for pSeries environment). Besides Emulex HBAs (http://www.emulex.com), in certain configurations you can use also Qlogic based adapters (http://www.qlogic.com). In this section we present such a configuration:

Download the appropriate drivers form HBA adapter's manufacturer Web site and unpack it in your ~/ directory. Then, perform the following:

```
cd /root/qlogic/i2x00-v7.01.01-fo/
```

Note: First remove the Red Hat QLogic drivers to make sure we are not mistakenly using them: (Not necessary for RHEL U4+)

```
rm -f /lib/modules/$(uname -r)/kernel/drivers/addon/*/qla2300*
```

Put the latest sources in ~/qlogic on node01 and patch the kernel header files (this is for improving driver performance) as shown in Example 4-12:

Example 4-12 Automating text replacement in qla2x00.h

```
perl -pi.orig -e '
      s|^(#define\s+SG_SEGMENTS)\s+32|$1 128|;
      s|^(#define\s+TEMPLATE_MAX_SECTORS\s+max_sectors:)\s+512,|$1 1024,|;
   ' qla2x00.h
```

Next build the module and install the driver options' altering command (**qla_opts**):

```
make clean all qla_opts install
cp -afv qla_opts /usr/local/sbin/
```

Copy the resulting kernel modules from node01 to other nodes

```
scp -rp /lib/modules/$(uname -r)/kernel/drivers/scsi/qla2300*.o
node02:/lib/modules/$(uname -r)/kernel/drivers/scsi/
...
```

Install **qla-autoconf**, change configuration and restart:

```
/root/qla-autoconf -r
```

Make sure you are using the correct driver and all the disks are available:

```
/root/qla-autoconf -v
#fdisk -l
cat /proc/scsi/scsi
tail -1000 /var/log/messages | grep -E 'sd[a-z]+'
```

Installation of the RDAC driver

The RDAC multipath is explained in detail in 4.1, “Linux multipathing” on page 140.

Start by making sure you don't have the failover driver, as shown in the following Example 4-13:

Example 4-13 Configuring and building the Qlogic module

```
cd /root/qlogic/i2x00-v7.01.01/
rm -f /lib/modules/$(uname -r)/kernel/drivers/addon/*/qla2300*
rm -f /lib/modules/$(uname -r)/kernel/drivers/scsi/scsi/qla2300*
perl -pi.orig -e '
      s|^(#define\s+SG_SEGMENTS)\s+32|$1 128|;
      s|^(#define\s+TEMPLATE_MAX_SECTORS\s+max_sectors:)\s+512,|$1 1024,|;
   ' qla2x00.h
make clean all qla_opts install
cp -afv qla_opts /usr/local/sbin/
```

Clear any qla2300 options in the modules.conf file. Clear also the qla2300_conf file.

```
rm -f /etc/qla2300.conf
touch /etc/qla2300.conf
/usr/loca/sbin/qla-opts --write qla2300_conf
```

Then build the RDAC driver:

```
cd /root/linuxrdac
make uninstall clean all install
```

Change lilo.conf or grub.conf (depending on your operating system) by adding a new stanza for booting with the new drivers, then run **lilo** and reboot using the new configuration. If everything works fine, copy the modules and configuration options to the remaining GPFS cluster nodes (see Example 4-14).

Example 4-14 Moving the modules from one system to another

```
NODE=node02
scp -rp /lib/modules/$(uname -r)/kernel/drivers/scsi/qla2300*.o
$NODE:/lib/modules/$(uname -r)/kernel/drivers/scsi/
scp -rp /lib/modules/$(uname -r)/kernel/drivers/scsi/mpp_*.o
$NODE:/lib/modules/$(uname -r)/kernel/drivers/scsi/
scp -rp /opt/mpp $NODE:/opt/
scp -rp /boot/mpp* $NODE:/boot/
scp -rp /var/mpp* $NODE:/var/
scp -rp /etc/mpp.conf $NODE:/etc/
scp -rp /usr/sbin/mpp* $NODE:/usr/sbin/
```

Configuring the GPFS Cluster

Refer to 3.4, “GPFS installation and configuration” on page 126.

4.5.1 Building gpfsperf

GPFS V2.3 provides a performance data collection tool, named **gpfsperf**. This tool has to be compiled on the system. To compile **gpfsperf** (assuming GPFS is already installed on your node) run this:

```
make -C /usr/lpp/mmfs/samples/perf clean all
```

On SLES9 ppc64 or x86_64 you may also need to run:

```
make -C /usr/lpp/mmfs/samples/perf clean all OTHERINCL=-m64
```

Then install the tool using:

```
install -Dp -m0755 /usr/lpp/mmfs/samples/perf/gpfsperf
/usr/local/bin/gpfsperf
```

4.5.2 Removing GPFS

To remove GPFS from a node, you need to check that no client is accessing any GPFS file system through that node, than you need to remove the node from the cluster. When removing the nod from the cluster you should check if the node does not have any special role (configuration data server - primary or secondary, quorum node, configuration manager, etc.) and that GPFS is stopped on the node.

Then proceed to removing the GPFS packages:

```
rpm -e gpfs.base gpfs.docs gpfs.gpl gpfs.msg.en_US
```

Remove GPFS and files:

```
rm -rf /var/adm/ras/ /var/mmfs/ /usr/lpp/mmfs/
```

If you removed GPFS from all nodes, you can check everything is uninstalled on all nodes (from node01) by using:

```
shmux -c "rpm -qa 'gpfs.*'" -<$SHMX
```

5

Maintaining and tuning the GPFS environment

In this chapter we present various tools and methods for maintaining and tuning your GPFS environment. Although not specifically tailored for digital media environments, similar tuning and management techniques apply to DM configurations. We also provide some guidelines about how to integrate GPFS with other software, and also some troubleshooting and performance tuning hints. The topics included in this chapter are:

- Performance tuning cycle
- Tuning the GPFS environment
- Software tools and utilities
- Client application considerations
- Performance tuning considerations
- List of GPFS related files

5.1 Performance tuning cycle

Tuning any kind of system requires a continuous review of all the system settings, an understanding of how the system acts under various circumstances, and access to performance data collecting tools. Tuning a GPFS cluster poses addition challenges, as the performance is not only determined by the system internals, but also by external elements, therefore requiring specific tuning techniques.

For every change you want to make, it is important to use a structured approach for analyzing the information and reaching the correct conclusions. Therefore, you need to prepare in advance the different tests you want to conduct (including the sequence they have to be performed), and document carefully the tests results (data collection).

There are a number of tools that can help you documenting the system configuration, monitoring performance counters and distributing/ verifying changes. A good understanding of these tools and the state of mind to optimize each of these steps will help you accelerate the process, will save you time afterwards to conduct additional tests, and will lead to more accurate test results in return.

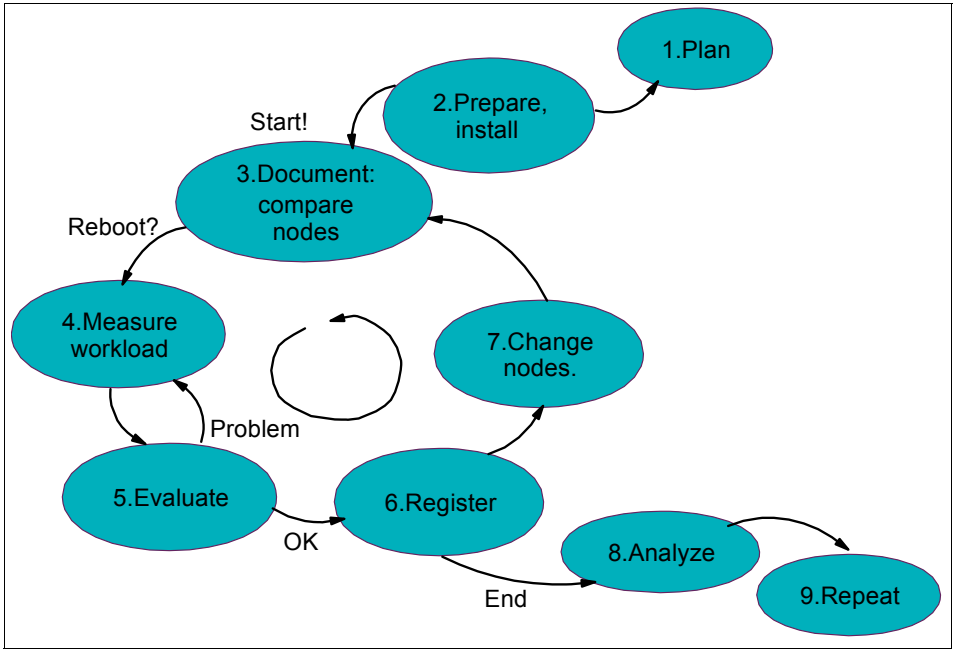

Figure 5-1 Performance tuning cycle

The following steps describe a performance tuning cycle, as depicted in Figure 5-1 on page 172:

1. **Plan**
   Before you start, think about the target you want to achieve and the possible ways of achieving that. Then think of what workload and conditions you want to test. If possible, use a real workload.

   Then logically break down the different parts of the system and look for aspects that qualify for being tested and/or changed.

   Make a list of all the changes you plan to conduct, think about combination of changes you want to test together and finalize a test plan.

   Add reference tests to your test plan, i.e., after a couple of tests, go back to your (initial) reference configuration, repeat the test and compare them with prior reference tests.

   Make a list of counters you want to register and look at tools that help you measure both the test workload as well as the performance of your system.

   Think of different ways to optimize and accelerate certain repetitive tasks and look at tools that can help you with that, doing this in advance will save time later.

2. **Prepare**
   Prepare the system(s) and the environment you are working in, make sure you have direct access to the most important performance counters and that you can evaluate the workload before, during and after the tests.

   Sync all the clocks on all the systems that are used to measure performance.

   Start your permanent monitoring tools.

3. **Document**
   Document the system configuration. If you have multiple identical nodes/devices, compare the configurations

4. **Measure**
   Reboot the system if necessary. If you believe that prior tests influence future tests, your results and the analysis based on that might be flawed.

   Start temporary monitoring tools, and write down exact time (time stamp).

   Start workload and evaluate workload counters during tests to catch failures during tests

   Stop temporary monitoring tools, and write down exact time (time stamp).

5. **Evaluate**
   Evaluate the workload based on performance counters and workload counters. If somehow you find something peculiar (e.g., identical nodes

behave differently, number of workload connections is too low), verify the configuration on all the systems or test setup and go back to step 4.

6. **Register**
   Create a directory and give it a distinct name. Copy the system configuration into this directory.

   Go over each of the monitoring tools, copy the output from the temporary monitoring tools and extract the appropriate performance data from the permanent monitoring tools. For the permanent monitoring tools you can do this step at the end too based on the timestamps.

   Add a description of what changes were made and report any problems or incidents that happened prior or during the test. They might not look relevant at the time.

   If something interesting comes up here, you can reconsider your test plan and add some additional tests to it if that seems wise. Don't remove tests from your test plan based on intermediate results though.

7. **Change**
   Perform a single change to your system. Never perform multiple changes at once unless you know exactly what you are doing. It is important for analysis later that you have as much data about isolated tests as possible to relate things.

8. **Analyze**
   Go over each of the saved test results and compare the results from isolated changes with those from combined changes to understand the results.

   Likely some of the results do not match with expectations or some of the results lead to a theory that needs more tests, that's why the analyze phase should be scheduled directly after finishing the tests so that extra tests can be performed.

   If your tests take multiple days, do an analysis each day and adapt your test plan for the next day to some of the findings.

9. **Report**
   Finalize your conclusions into a report that includes your projected target, your workload structure, your test plan, and, for each test, the system configuration, the performance data and your analysis.

5.2 Tuning the GPFS environment

GPFS performance depends on the correct use of its tuning parameters as well as the tuning of the infrastructure it relies on. It is important to understand that the following guidelines are "as is" and there is no guarantee that applying these guidelines will result in a performance improvement in your environment.

Tuning parameters depend on a given workload (generated by real applications), the overall design, and on understanding the infrastructure (hardware, storage, networking, operating system). Therefore it is important to measure the impact of changes as thorough as possible under similar or reproducible load, using the infrastructure in the production environment.

Attention: Since it is practically impossible to reproduce behavior of client applications and appliances using load-generation or benchmark tools, it is generally advised to measure performance changes caused by changing tuning parameters in the actual customer environment, preferably under circumstances as close as possible to the real production situation.

It is equally important to implement a change management and a fallback plan in order to be able to bring the cluster back to the previous state every time you perform a test.

5.2.1 Tuning storage

As we have used the IBM TotalStorage DS4500, we suggest checking and adjusting the following parameters at the storage level:

- RAID5 arrays with five disks using a 4+P scheme.
- Cache block size: 16 KB
- Configuration for **data** LUNs:
  - Segment size: 256 KB (to match the GPFS block size of *4 x 256kB = 1MB*)
  - Read cache disabled
  - Write cache, write cache mirroring, and write cache without batteries are all disabled
  - Modification priority: low
- Configuration for **metadata** LUNs:
  - Segment size: 64 KB
  - Read cache enabled
  - Read ahead multiplier: 5
  - Write cache, write cache with mirroring, and write cache without batteries are all enabled. Make sure the batteries are working properly.
  - Modification priority: low

Attention: If the data availability requirement is not critical, write performance can be improved significantly by enabling write caching. The drawback is that the DS4500 maintains a volatile disk cache distributed between two distinct RAID controllers. When a write operation is completed (including flushing buffers), control is returned to the application when the data is written to this cache, but not yet committed to the disk.

If one of these RAID controllers fails, then the data in the associated cache will be lost; since this can include metadata, the entire file system can be corrupted. Given the potentially large size of GPFS file systems (for example, 100s of terabytes or more) and the fact that files are striped over all of the disks, the magnitude of the risk is multiplied.

Enabling write cache mirroring can compensate for this risk, but may compromise storage performance associated with write caching, because the cache synchronization takes place on the backend disk loops (there is no dedicated bus for this operation).

For high availability, the disks of a particular LUN should be distributed over as many disk enclosures as possible. Ideally, every enclosure should contain only one (1) disk of each RAID array. For example, for 4+P arrays you should use at least five expansion units and configure the array by using one drive in each expansion unit.

This will protect from outages in case a complete expansion unit fails. In addition, the expansion units are cabled in such a way that two independent loops will have access to the same drives (see Figure 3-5 on page 91 for 4+P array configuration and Figure 3-6 on page 92 for back end cabling).

5.2.2 Tuning the SAN hardware

While tuning a SAN environment may be an undertaking not generally accessible to everyone, some SAN elements can be checked and adjusted by local system administrators. We present some basic guidelines for tuning your SAN. Depending on the complexity on your environment and the equipment characteristics, vendor supplied GUI tools can be used to perform theses tasks.

Note: It is generally NOT recommended to mix hardware from different vendors into the same SAN. As each vendor has implemented specific functionality, you cannot benefit of the full functionality as you will need to configure all devices in compatibility mode, which dramatically decreases SAN configurability, flexibility, and even performance.

The most important aspects you need to check at the SAN level are:

- All ports must be configured to the same speed (2 Gbit/s).
- Implement proper zoning: every HBA should have its own zone, as shown in Figure 5-2.

  Zoning is a SAN security feature which prevents unauthorized access from hosts to LUNs they are not entitled to access.

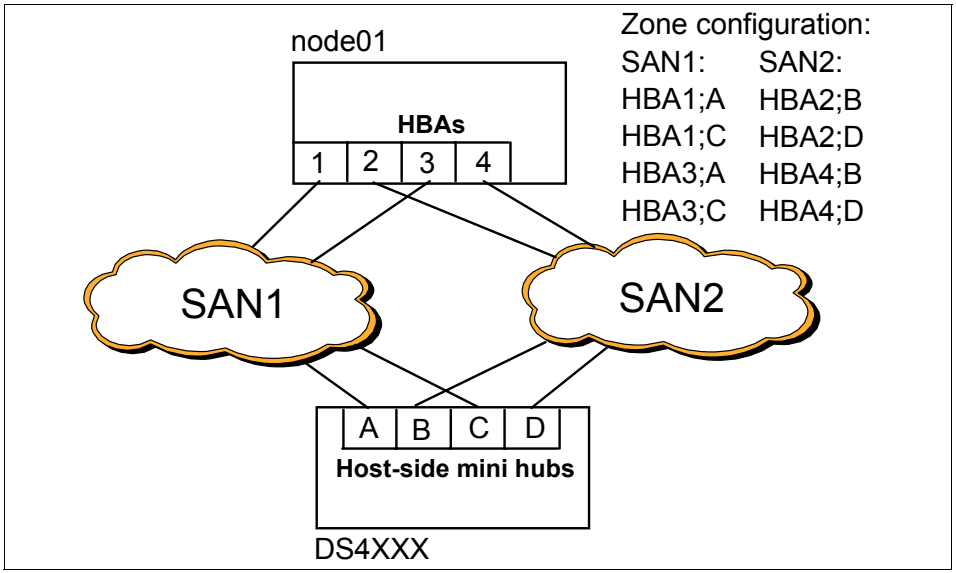

Figure 5-2 Sample zoning (one host only)

Tuning the HBA parameters

In certain cases you may need to tune the HBA driver parameters. You should follow the vendor documentation and/or additional product documentation. We do not cover HBA tuning in this section. In most cases though, the HBA drivers are designed to auto detect and auto configure most parameters.

5.2.3 Tuning the operating system

Tuning AIX

For tuning AIX, you should follow the documentation provided, specially the *AIX Performance Management Guide*, SC23-4905. You can find the online version of this manual at:

http://publib.boulder.ibm.com/infocenter/pseries/index.jsp?topic=/com.ibm.aix.doc/aixbman/prftungd/prftungd.htm

The two impacting tuning areas for a GPFS cluster are network (layers 2, 3, and 4) and NFS parameters (specially if you use NFS exported GPFS file systems).

Tuning Linux

Because there is a lot of unstructured information about tuning Linux, in the following section we have put together some basic tuning tools and parameters for Linux environments.

You should keep in mind that there are a lot of tools and techniques out there, but performance tuning has always been more of an art that an exact science, therefore the information presented here is not exhaustive, nor the single way to achieve results.

Linux kernel tuning parameters and the /proc file system

The /proc file system allows direct access to system information and tunable parameters inside the kernel. Most of the files in the /proc file system are read-only, but a selection (mostly inside /proc/sys) can be altered. This allows you to change the behavior of the kernel instantly.

Displaying and changing values in /proc/sys can simply be done on the command line, as shown in Example 5-1:

Example 5-1 Checking and altering the proc values

```
node01:~ # cat /proc/sys/kernel/sysrq
0
node01:~ # echo -n "1" > /proc/sys/kernel/sysrq
```

However, these changes are not permanent and the values will revert back to their default after rebooting the system. For that reason it is more simple to use the **sysctl** interface and the `/etc/sysctl.conf` configuration file to display and change the kernel tunable parameters permanently.

For example, if we want to see the maximum number of processes on the local system, you can use:

```
node01:~ # sysctl kernel.pid_max
kernel.pid_max = 32768
```

If you want to make a temporary change, you can use **sysctl** as follows:

```
node01:~ # sysctl -w kernel.pid_max="65536"
kernel.pid_max = 65536
```

The proper way to make a permanent change is to add an entry to the /etc/sysctl.conf file.

```
kernel.pid_max = 65536
```

The **sysctl** command also allows you to see the kernel parameters, as shown in Example 5-2:

Example 5-2 Displaying kernel parameters (excerpt)

```
node01:~ # sysctl -a
......>>> Omitted lines <<<........
sunrpc.tcp_slot_table_entries = 16
sunrpc.udp_slot_table_entries = 16
sunrpc.nlm_debug = 0
sunrpc.nfsd_debug = 0
......>>> Omitted lines <<<........
```

Note: SLES distributions do not, by default, run **sysctl** on boot. To activate **sysctl** changes each time the system boots, use:

```
chkconfig boot.sysctl on
```

In case you want to browse over the list, you might want to sort the output and pipe it through less in true Unix fashion:

```
node01:~ # sysctl -a | sort | less
```

General kernel tuning parameters

- **kernel.core_uses_pid** - enable/disable use of pid in core files, disable this if your filesystem might be trashed by core files
- **kernel.pid_max** - maximum number of processes, increase if necessary
- **kernel.shmmax** - maximum amount of memory available to /dev/shm (tmpfs)
- **kernel.sysrq** - enable/disable the kernel 'sysrq' functionality

Table 5-1 Overview of useful tuning parameters

Sysctl variable	Default 2.6	Suggested
kernel.pid_max	32768	32768*
vm.lowmem_reserve_ratio	256 32	
vm.overcommit_ratio	50	
vm.swapiness	60	

Network tuning parameters

Tuning the following parameters will modify the TCP window sizes for the Read and Write buffers for the network adapters.

Table 5-2 Overview of useful network tuning parameters

Sysctl variable	Default 2.6	Suggested
net.ipv4.conf.all.arp_filter	0	1
net.ipv4.conf.all.rp_filter	0	1
net.ipv4.tcp_max_syn_backlog	1024	4096
net.ipv4.ipfrag_low_thresh	196608	262144
net.ipv4.ipfrag_high_thresh	262144	393216
net.core.netdev_max_backlog	300	2000
net.core.rmem_default	135168	262144
net.core.rmem_max	131071	8388608
net.core.wmem_default	135168	262144
net.core.wmem_max	131071	8388608
net.ipv4.ip_local_port_range	32768 61000	1024 65535
net.ipv4.ip_no_pmtu_disc	0	0
net.ipv4.tcp_rmem	4096 87380 174760	8192 262144 4194304
net.ipv4.tcp_sack	1	0
net.ipv4.tcp_syncookies	1	0
net.ipv4.tcp_timestamps	1	0
net.ipv4.tcp_window_scaling	1	1
net.ipv4.tcp_wmem	4096 16384 131072	8192 262144 4194304

Tuning the network hardware

In addition to TCP/IP network parameters, you can improve the network performance by using one of these configuration options of a combination:

- Ethernet jumbo frames
  The default Maximum Transmission Unit (MTU) for Ethernet network is 1500 bytes. This has to accommodate the protocol headers (IP, TCP/UDP) and the useful information. By using jumbo frames you can increase the size of the MTU, therefore allowing larger data blocks to be transferred in a single step. This improves performance for certain applications that use large blocks of data by providing for less data fragmentation.

For configuring jumbo frames you need network elements that support this option, from adapter and device driver to Ethernet switch. Not all switches support this feature, but generally, high end switches do so.

- Network bonding
  Beside jumbo frames, ethernet bonding is another technique that allows for improved network performance. There are several implementations to bond network interfaces:
  - Link aggregation - IEEE 802.3ad
  - CISCO Etherchannel
  - Linux bonding

  The principle is the same: putting together two or more ethernet adapters in the same node to form a single communication channel that allows for aggregate bandwidth, high availability, or both.

  A good starting point for understanding the various ways to configure Ethernet bonding is the Linux kernel documentation (in the kernel source).

5.2.4 Tuning GPFS

This section provides information about GPFS tuning parameters. GPFS tuning parameters can influence the behavior at cluster level, node level, or file system level. A good information source for tuning GPFS (although mostly for reference) in the redbook "*Sizing and Tuning GPFS*", SG24-5610.

Important: Make sure that you have enough physical resources in your node whenever you plan to modify certain GPFS parameters. For example, if you modify “pagepool” for your node, if there is not enough physical memory, the node will not start, and this failure may not be obvious.

GPFS configuration options

Once the GPFS cluster has been configured, you can issue the following commands before creating any file system (using the **mmcrfs** command) to change the default cluster parameters, then create the GPFS filesystem with the following command with a block size of 1024K (Example 5-3):

Example 5-3 Changing the cluster default parameters

```
# mmchconfig  maxblocksize=1024k
# mmchconfig  pagepool=2000M
# mmchconfig  maxMBps=800
# mmchconfig  tiebreakerDisks="gpfsNNnsd;gpfsNNnsd;gpfsNNnsd"
......
You should replace NN with the appropriate generated number for the nsd disk
.....
```

```
# mmcrfs /gpfs /dev/gpfs -F <disk descriptor file> -A no -B 1024k -E no  -m 1
-M 1 -r 1 -R 1 -v
```

mmfs.cfg

The /var/mmfs/etc/mmfs.cfg file can be edited to contain configuration options for all your nodes. You can define common options and then override these on a per node or per nodelist basis. This allows to centrally manage the file and distribute it to all the nodes. A sample mmfs.cfg file is shown in Example 5-4.

Note: The mmfs.cfg file (in /var/mmfs/etc) is generated once you configure the cluster the first time. You can modify it afterwards.

Example 5-4 Modifying the default GPFS parameters in mmfs.cfg

```
## Sample configuration file

## Numbers may end with a single letter:
##   k or K meaning 1024
##   m or M meaning 1048576 (1024*1024)
##
## The '#' character is the comment character.  Any parameter
## modified herein should have any preceding '#' removed.
##

########  Memory / Shared Segment Configuration  ########

## The pagepool is used for I/O buffers.  It is always pinned.
## The allowable range is 4M to 512M (AIX).
## The allowable range is 4M to 1300M (LINUX).
#pagepool 64M

## maxblocksize controls the maximum file system block size allowed.
## File systems with larger block sizes cannot be mounted or created
## unless the value of maxblocksize is increased.
## The allowable range is 16K to 16M
## default: maxblocksize 1M
#maxblocksize

## Maximum number of files to cache. If the number of concurrently open
## files is bigger, then the number of cached files will exceed this value.
## The allowable range is 1 to 100000
#maxFilesToCache 1000

## The maximum number of stat cache entries.
## The default is 4 times the value of the maxFilesToCache parameter.
## The allowable range is 0 to 10000000
#maxStatCache
```

```
########  DMAPI configuration  ########

## The dmapiEventTimeout parameter controls the blocking of file operation
## threads of NFS and DFS, while in the kernel waiting for the handling of
## a DMAPI synchronous event. The parameter value is the maximum time, in
## milliseconds, the thread will block. When this time expires, the file
## operation returns ENOTREADY, and the event continues asynchronously.
## The NFS/DFS server is expected to repeatedly retry the operation, which
## eventually will find the response of the original event and continue.
## This mechanism applies only to read, write and truncate events, and only
## when such events come from NFS and DFS server threads. The timeout value
## is given in milliseconds. The value 0 indicates immediate timeout (fully
## asynchronous event). A value greater or equal 86400000 (which is 24 hours)
## is considered "infinity" (no timeout, fully synchronous event).
## The default value is 86400000.
#dmapiEventTimeout 86400000

## The dmapiSessionFailureTimeout parameter controls the blocking of file
## operation threads, while in the kernel, waiting for the handling of a DMAPI
## synchronous event that is enqueued on a session that has suffered a failure.
## The parameter value is the maximum time, in seconds, the thread will wait
## for the recovery of the failed session. When this time expires and the
## session has not yet recovered, the event is aborted and the file operation
## fails, returning the EIO error. The timeout value is given in full seconds.
## The value 0 indicates immediate timeout (immediate failure of the file
## operation). A value greater or equal 86400 (which is 24 hours) is considered
## "infinity" (no timeout, indefinite blocking until the session recovers).
## The default value is 0.
#dmapiSessionFailureTimeout 0

## The dmapiMountTimeout parameter controls the blocking of mount operations,
## waiting for a disposition for the mount event to be set. This timeout is
## activated at most once on each node, by the first external mount of a
## file system which has DMAPI enabled, and only if there has never before
## been a mount disposition. Any mount operation on this node that starts
## while the timeout period is active will wait for the mount disposition. The
## parameter value is the maximum time, in seconds, that the mount operation
## will wait for a disposition. When this time expires and there is still no
## disposition for the mount event, the mount operation fails, returning the
## EIO error. The timeout value is given in full seconds. The value 0 indicates
## immediate timeout (immediate failure of the mount operation). A value
## greater or equal 86400 (which is 24 hours) is considered "infinity" (no
## timeout, indefinite blocking until the there is a disposition).
## The default value is 60.
#dmapiMountTimeout 60

########  Prefetch tuning ##########
```

```
## The value of the 'prefetchThreads' parameter controls the maximum
## possible number of threads dedicated to prefetching data for
## files that are read sequentially, or to handle sequential writebehind.
## The actual degree of parallelism for prefetching is determined
## dynamically in the daemon.
## (minimum 2, maximum 104)
#prefetchThreads 72

## The 'worker1Threads' parameter controls the maximum number of threads
## that are used to handle other operations associated with data access.
## The primary use is for random read/write requests that cannot be prefetched.
## random IO requests or small file activity.
## (minimum 1, maximum 64)
#worker1Threads 48

## maxMBpS is an estimate of how many MB per sec of data can be transferred
## in or out of a single node. The value is used in calculating the
## amount of IO that can be done to effectively prefetch data for readers
## and/or or write-behind data from writers. The maximum number of IOs in
## progress concurrantly will be 2 * min(nDisks, maxMBpS*avgIOtime/blockSize),
## where nDisks is the number disks that make a filesystem,
## avgIOtime is a measured average of the last 16 full block IO times, and
## blockSize is the block size for a full block in the filesystem (e.g. 256K).
## By lowering this value, you can artificially limit how much IO one node
## can put on all the VSD servers, if there are lots of nodes that
## can overrun a few VSD servers. Setting this too high will usually
## not hurt because of other limiting factors such as the size of the
## pagepool, or the number of prefetchThreads or worker1Threads.
#maxMBpS 150

########  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 'unmountOnDiskFail' keyword controls how the daemon will respond when
## a disk failure is detected.
##
## When it is set to "no", the daemon will mark the disk as failed and
## continue as long as it can without using the disk. All nodes that are
## using this disk will be notified of the disk failure.  The disk can be
## made active again by using the "mmchdisk" command.  This is the
## suggested setting when metadata and data replication is used because
## the replica can be used until the disk can be brought online again.
##
## When this is set to "yes", any disk failure will cause only the local
## node to panic (force-unmount) the filesystem that contains that disk.
## Other filesystems on this node and other nodes will continue to function
```

```
## normally (if they can. The local node can try and remount the filesystem
## when the disk problem has been resolved. This is the suggested setting
## when using VSD disks in large multinode configurations and replication is
## not being used.
#unmountOnDiskFail no

## 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

######## 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.  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]
##     maxFilesToCache 200
##
##   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, i.e. every node will have a maxFilesToCache
##   of 200.
```

5.2.5 Tuning the clients

In a DM environment most of the clients are Windows based, and to obtain the best results you also need to tune the network subsystem on the client side. This section contains recommendations on Windows network tuning in a DM environment.

Tuning Windows clients

Here are a few tips for improving the network subsystem configuration on Windows:

- Set the Windows memory strategy to “Maximize Throughput for Network Applications” if applicable (see Control Panel → Network → Server).
- Disable all unneeded services, protocols, protocol bindings and drivers.
- Set the network adapter receive and transmit buffers to an optimal performance. (See Table 5-3 for the Intel Pro 1000 adapter for values.)

Table 5-3 Table: Windows tuning parameters for Intel Pro 1000 gigabit adapter

Registry parameter	Default	Suggested	Description
NumberOfReceiveBuffers	200	768	Number of posted receive buffers on the adapter
NumberOfCoalesceBuffers	200	512	Number of buffers before an interrupt occurs
NumberOfReceiveDescriptors	448	512	Number of receive buffer descriptors on the adapter
NumberOfTransmitDescriptors	448	512	Number of transmit buffer descriptors on the adapter
ReceiveChecksumOffloading	Off	On	Task offloading onto hardware in receiver path
TransmitChecksumOffloading	On	On	Task offloading onto hardware in sender path

- Balance client I/O among network adapters.
- Enable TCP checksum offloading for you network adapter, if it supports it.
- Disable performance boost for foreground applications (see Control Panel → System → Advanced → Performance Options).
- Optimize pagefile by creating a fixed size page file that is at least 2 times the amount of RAM available (preferably on a dedicated or separate disk drive).
- Tune TCP/IP registry values to improve network performance (see table with common registry values).

- Tweak process scheduling, priority levels and affinity (use Task Manager to see values and change priority).
- Assign interrupt affinity for hardware devices (use the intfiltr utility provided with Windows).
- If necessary use the /3GB BOOT.INI parameter to allocate 3 GB to user space instead of 2 GB for user space and 2 GB for kernel space.

Table 5-4 Table: Common Windows registry settings

Registry parameter	Default	Suggested	Description
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\TcpWindowSize	0x4470 (17250)	0xFAF0 (62420)	Larger TCP receive window size will improve performance over high-speed networks (tweak and test for your environment)
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\MaxFreeTcbs	200	0xFA0	Timewait table size
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\MaxHashTableSize	0x200 (512)	0x2000 (8192)	TCP Control Block (TCB) hash table size
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\NumTcbTablePartitions	n ^ 2	4 * n	Number of TCB tables partitions (n is the number of processors) value should be power of 2
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\TCP1323Opts	0x0	0x3	Enable large TCP windows scaling and RTT (timestamps) estimation
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Interface\XXX\TcpAckFrequency	0x2	0x5 or 0xD	Delayed acknowledgement frequency for TCP (0x4 for 100Mbit or 0xD for Gigabit)
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\Interface\XXX\MTU	0xFFFFFFFF	1450 or 9000	0xFFFFFFFF means a dynamic MTU, needs to be fixed if we disable PMTU
HKLM\System\CurrentControlSet\Services\Tcpip\Parameters\EnablePMTUDiscovery	1	0	Disable MTU path discovery
HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\Size	-	3	Optimize for maximum data throughput for network applications

Registry parameter	Default	Suggested	Description
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\LargeSystemCache	-	0	Optimize for maximum data throughput for network applications
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\DisablePagingExecutive	0x1	0x0	Disable kernel paging when our clients have enough memory
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\PagedPoolSize	0x0	0xB71B000 (192000000)*	0xFFFFFFFF means dynamically created by Windows, the example creates a 192MB pagepool, which means a virtual address space of 960MB
HKLM\System\CurrentControlSet\Control\Session Manager\Memory Management\SystemPages	0x0	0x0	Required for customizing the PagedPoolSize
HKLM\System\CCS\Services\LanmanServer\Parameters\MaxWorkitems	-	65535	Maximum number of receive buffers that CIFS is permitted to allocate at one time
HKLM\System\CCS\Services\LanmanWorkstation\Parameters\MaxCmds	50	4096	Maximum number of network control blocks that the redirector can reserve
HKLM\System\CCS\Services\LanmanServer\Parameters\MaxMpxCt	-	2048	Maximum number of simultaneous outstanding requests on a client to a particular server

For more information about Windows performance tuning, see the redbook *Tuning IBM @server xSeries Servers for Performance*, SG24-5287.

5.3 Software tools and utilities

This section contains a short introduction to various tools and utilities available for system administration and performance tuning in a UNIX clustering (and not only) environment. Usually, the best tools you can use are the ones you are familiar to, but it is a good idea to check periodically for new or updated tools that may help you solve administrative problems quicker and safer.

Depending on the platform and distribution (AIX, SLES, Red Hat, etc.) the available tools may differ, but there is a set of tools widely available on all UNIX (and similar) platforms.

General UNIX tools: `iostat, vmstat, netstat, tcpdump, iozone, sar, rrdtool, ps, time, nmon, nfsstat, netpipe, ttcp, nttcp, mrtg, cacti, mmpmon, netperf, ganglia, gpfsperf, mmtrace, ifconfig, dd and dd_rescue, top, mpstat, iperf`

There are also platform specific tools for AIX, Linux, and Windows:

AIX: `truss, svmon, lslv, fileplace, filemon, iptrace, netpmon, entstat, lsps, estat, statsvsd`

Linux: `ifstat, strace, ltrace, ethereal, stress, tiobench, iftop, dstat, ethtool, iometer, pmap`

Windows: `cygwin, dd, iozone, filemon, regmon, iperf`

Table 5-5 contains a classification of the tools available.

Table 5-5 Classification of tools

command	Linux	AIX	Windows	system resources	network	disk/ filesystem	application
System administration tools							
dconf	X						
rdist	X	X					
scp	X	X	X				
shmux	X	X					
screen	X	X					
watch	X	X					
yam	X						
System resource monitoring tools							
dstat	X			X	X	X	
ifstat	X				X		
iostat	X	X				X	
mmpmon	X	X				X	
mpstat	X	X		X			
netstat	X	X			X		

command	Linux	AIX	Windows	system resources	network	disk/ filesystem	application
nfsstat	X	X			X		
nmon	X	X		X	X	X	X
sar	X	X		X		X	
top	X	X		X			X
vmstat	X	X		X		X	
Load generation tools							
dd / dd_rescue	X	X	X			X	
gpfsperf	X	X				X	
iometer	X	?				X	
iozone	X	X	X			X	
iperf	X	?	X		X		
netperf	X	?			X		
netpipe	X	X			X		
spew	X					X	
stress	X			X		X	
ttcp / nttcp	X				X		
Debugging tools							
ethereal	X	X	X		X		
filemon		X	X			X	X
laus	X						
lsof	X	X				X	X
ltrace	X	X				X	X
mmtrace	X	X				X	
oprofile	X						
pmap	X	X		X			

command	Linux	AIX	Windows	system resources	network	disk/ filesystem	application
regmon			X				X
strace	X					X	X
tcpdump	X	X			X		X
truss		X					X
Graphing tools							
cacti	X						
ganglia	X						
ksysguard	X						
mrtg	X	X	X				
perfmon			X				
rrdtool	X	X					

5.3.1 System administration tools

This section describes the system administration tools.

dconf - system configuration management

Dconf is a tool written in python that allows you to make a snapshot of the complete configuration of a Linux system. This snapshot can then be collected to compare with other systems.

Dconf can be configured to automatically make snapshots at certain intervals (every hour, day, week or month) and if necessary send out e-mail containing the changes since the last snapshot. This makes Dconf useful as a change management tool for any environment where you have to keep track of changes. Dconf will not keep snapshot files that are identical to the latest snapshot, so that you have a chronological view of changes with a minimum of resources.

Dconf can be useful to track and verify changes during performance tuning or troubleshooting process, as it provides a good level of control with a minimum effort.

The default Dconf configuration file also contains predefined commands and configuration files for a GPFS environment, so no special adjustments are necessary to use it for a GPFS cluster.

Here is a (short) example of how one can use it for comparing system configurations using **shmux** and **diff**:

```
node01:/var/log/dconf # shmux -c "dconf" -</etc/gpfs/shmux
3 targets processed in 5 seconds.
Summary: 3 successes
```

Dconf has created a snapshot on each system and has a symbolic link pointing to the latest snapshot (/var/log/dconf/dconf-node0X-latest.log.gz). We can copy these files to one system and compare the files locally (Example 5-5):

Example 5-5 Gathering the dconf information

```
host:~ # scp -rp node01:/var/log/dconf/*-latest.log.gz .
host:~ # scp -rp node02:/var/log/dconf/*-latest.log.gz .
host:~ # scp -rp node03:/var/log/dconf/*-latest.log.gz .
host:~ # zdiff -u dconf-node01-latest.log.gz dconf-node02-latest.log.gz | less
```

Or we can compare the logfile on node01 (the current node) and node02 (using ssh):

```
node01:/var/log/dconf # diff -u <(zcat *-latest.log.gz) <(ssh node02 zcat
$PWD/*-latest.log.gz) | less
```

Note: If you see several differences that are not related to configuration changes, make sure that each system has the same list of packages installed in an identical manner so that differences are reduced to a minimum.

However, in case that is not an option, you can also create your own minimal configuration file and run dconf with the -c option to reduce the differences.

You can configure Dconf to run periodically (every hour, day, week, month) and it will store a timestamped snapshot file in /var/log/dconf, but only if it differs from the previous run. If you want Dconf to send e-mail for every change, add the following to /etc/dconf-custom.conf and run Dconf once.

```
[main]
mailto = email@address.com, email2@address.com
cron = weekly
```

More information about Dconf and packages for different distributions, including Red Hat and SUSE LINUX can be found at the Dconf Web site:

http://dag.wieers.com/home-made/dconf/

rdist - remote file distribution client program

The **rdist** tool helps you maintain identical files on multiples nodes. It does this by overwriting the files on every node by those on a designated master server.

There are two ways of using **rdist**, either by manually running **rdist** when files have been altered, or by running it periodically (e.g., from **cron**). Upon execution **rdist** will read the provided configuration file, connect to remote nodes via **ssh** or **rsh**, compare the files, and in case they are different, update them.

The **rdist** tool expects an input file (distfile) that describes the list of nodes (excluding the local one), a list of files and/or directories that need to be synchronized, and additional flags or commands to run on the remote node in case special files have changed (see Figure 5-3).

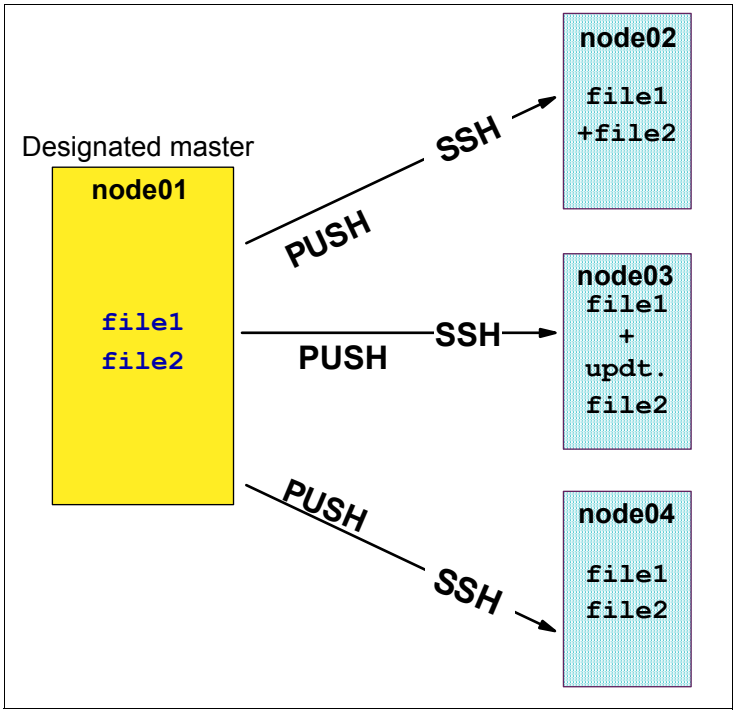

Figure 5-3 Using rdist to update files

We present here in Example 5-6 a sample distribution list we have used on our designated master node01 (stored in /etc/gpfs/rdist):

Example 5-6 Rdist distribution file

```
HOSTS = ( node02 node03 )
```

```
FILES = (
        /etc/gpfs/shmux
       /root/.ssh/id_dsa
        /root/.ssh/id_dsa.pub
        /root/.ssh/authorized_keys
        /root/.bashrc
)

${FILES} -> ${HOSTS}
        install -savetargets;
```

To verify the status of the files managed by **rdist**, you can use the -v option (Example 5-7):

Example 5-7 Checking the rdist status

```
node01:~ # rdist -v -f /etc/gpfs/rdist -P /usr/bin/ssh
node02: updating host node02
node02: /root/.ssh/authorized_keys: need to update
node02: updating of node02 finished
node03: updating host node03
node03: /root/.ssh/authorized_keys: need to update
node03: /root/.bashrc: need to install
node03: updating of node03 finished
```

And to update the files, use (Example 5-8):

Example 5-8 Manual update using rdist

```
node01:~ # rdist -f /etc/gpfs/rdist -P /usr/bin/ssh
node02: updating host node02
node02: /root/.ssh/authorized_keys: updating
node02: updating of node02 finished
node03: updating host node03
node03: /root/.ssh/authorized_keys: updating
node03: /root/.bashrc: installing
node03: updating of node03 finished
```

For the **rdist** to run properly, you need passwordless communication using **ssh** or **rsh**. For more details on **ssh** key exchange see 4.2, “SSH key distribution” on page 155.

For more information, see the homepage of the rdist project at:

http://www.magnicomp.com/rdist/

A good introduction to rdist can also be found at:

http://www.benedikt-stockebrand.de/rdist-intro.html

scp - remote secure copy

scp is part of the **ssh** toolkit and allows to copy files between different systems much like **rcp**, but using secure communication (password and data encrypted using ssh keys). If you already set up passwordless communication using **ssh**, **scp** will use the same authentication.

In this example we copy a file from the current node (node01) as the same user (root) to node02 in the user's home directory:

```
node01:~ # scp rdac_LINUX_09.01.B5.02.tar.gz node02:
```

Alternately, you can use:

```
node01:~ # scp rdac_LINUX_09.01.B5.02.tar.gz root@node02:~
```

scp is not useful for distributing files to a large number of systems, but together with **shmux** (or **dsh)** it allows you to pull individual files from a system, and distribute them to other systems:

```
node01:~ # sshmux -c "scp node01:rdac_LINUX_09.01.B5.02.tar.gz ." node02
node03
```

For more information about how to use shmux, see "shmux - shell multiplexer" on page 196.

screen - tty multiplexer

The **screen** tool is often described as a tty multiplexer. It offers you multiple virtual terminals over a single connection and provides you with key shortcuts to switch between these terminals. It is very useful for managing and monitoring systems as it provides you with an easy way of using multiple shell screens with a minimum of effort.

It is also particularly useful if you have jobs running that either cannot be interrupted, you need to occasionally interact with them, or you want to see their output at a later time.

And a third usage pattern for **screen** is if you want to share a single shell between different users (on different remote locations).

To use **screen**, you have to logon to a remote system using **ssh** or **rsh** to access the user's default shell. Once you are logged in, to start a new session inside the same screen, just use the **screen** command, and you will start the virtual screen 1.

To switch back to the virtual terminal 0, use *<ctrl-a> 0*, and for virtual terminal 1 use *<ctrl-a> 1*.

Detaching from and re-attaching to a terminal

Next, we describe the detach and re-attach functionality. Use the virtual terminal 0 and type: **`vmstat 1`** (this will generate continuous output). Now type *<ctrl-a> d* to detach from this screen. You will see the original shell that you got when you logged on to this system. Log out from the system. The **`screen`** (and **`vmstat`**) are still running in the background. To check this, log back on to the system and type: **`screen -x`** (attach to the existing screen). You should see the **`vmstat`** still running on virtual terminal 0.

Sharing a terminal between remote users

As you still have **`vmstat`** running in virtual terminal 0, open a second **`ssh`** connection (preferably from another system) and log on to the system running **`screen`**. Now type **`screen -x`** (attach to the existing screen). You will see the **`vmstat`** running on virtual terminal 0 (it may require you to switch to virtual terminal 0 first). Check both ssh connections you have open, and stop **`vmstat`**. If Now you type in one of the terminals, you will see the same activity in the remote terminal.

Using screen for education and demonstration

This functionality is useful in different scenarios — for example, when you have to demonstrate a problem to the development team on the other side of the ocean, or if a customer has problems and you want to show them how to fix the problem, or if a junior system administrator, or any untrusted individuals, has to perform an activity and you want to only allow supervised root access.

shmux - shell multiplexer

The **`shmux`** tool allows you to run commands on a group of nodes in parallel. It uses **`ssh`** (or **`rsh`**) for each spawned process. The output is display in an easy to read format (prepended by the node name). It is similar in functionality with the distributed shell (**`dsh`**) that is supplied with IBM's Cluster Systems Management software.

By default, **`shmux`** first runs the command on one node, and if command is successful, it proceeds with the other nodes in parallel (see Example 5-4 on page 197). It highlights the standard error output, shows progress in real time (unless specified otherwise), it automatically pauses on error, allows to control progress (pause, resume, quit) when required, and has special scripting capabilities.

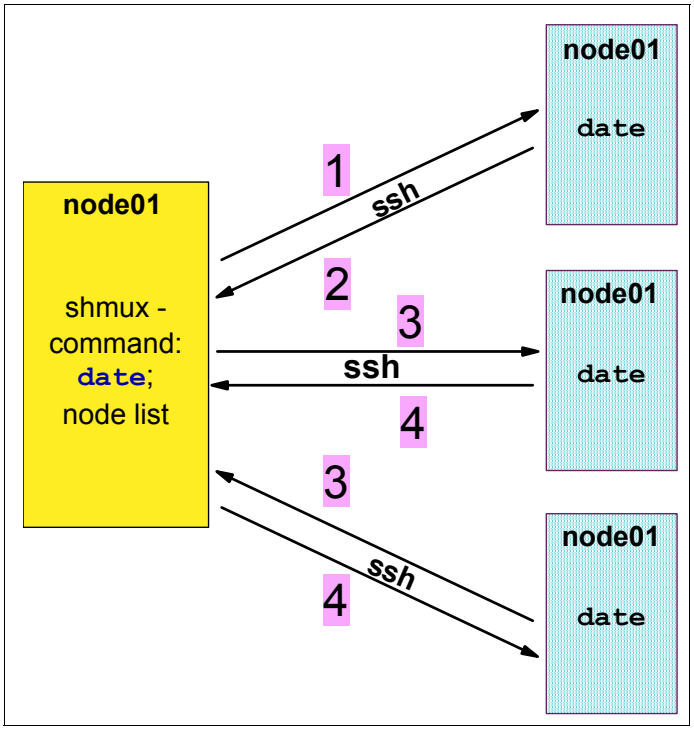

Figure 5-4 shmux operation

shmux requires passwordless communication using **ssh** or **rsh**. For details about how to configure ssh for passwordless operation, see 4.2, “SSH key distribution” on page 155.

Example 5-9 shows how you can use shmux:

Example 5-9 Using shmux

```
node01:~ # shmux -c 'date' node01 node02 node03
node01: Tue Jun  7 01:20:06 EDT 2005
node02: Tue Jun  7 01:20:07 EDT 2005
node03: Tue Jun  7 01:20:16 EDT 2005

3 targets processed in 1 second.
Summary: 3 successes
```

It is generally a good idea to keep the cluster nodes in a list (file). This makes it easier to execute commands on all cluster nodes without having to specify them on the **shmux** command line. You can group systems together in different files if

you plan to run commands on subsets of nodes. See Example 5-10 for a sample node list file:

Example 5-10 Sample node list file

```
node01:~ # cat /etc/gpfs/shmux
node01
node02
node03
```

And use it as shown in Example 5-11:

Example 5-11 Using the node list file

```
node01:~ # shmux -c 'grep gpfs /proc/mounts' -</etc/gpfs/shmux
node01: /dev/gpfs /gpfs gpfs rw 0 0
node02: /dev/gpfs /gpfs gpfs rw 0 0
node03: /dev/gpfs /gpfs gpfs rw 0 0

3 targets processed in 1 second.
Summary: 3 successes
```

You can also export the file as an environment variable (e.g., SHMX, see Example 5-12):

Example 5-12 Exporting the node list file as a variable - SHMX

```
node01:~ # export SHMX=/etc/gpfs/shmux
node01:~ # shmux -c 'rpm -q gpfs.base' -<$SHMX
node01: gpfs.base-2.3.0-3
node02: gpfs.base-2.3.0-3
node03: gpfs.base-2.3.0-3
```

To make the variable permanent for root user, ad it to ~/.bashrc.

If something goes wrong, you will see these messages (Example 5-13):

Example 5-13 shmux error messages

```
node01:~ # shmux -c false -<$SHMX
 shmux! Child for node01 exited with status 1
-- [PAUSED], 1 Pending/0 Failed/1 Done -- [0.0, 0.0]
```

If you press Enter to continue with the other nodes, you will see the following messages (Example 5-14):

Example 5-14 shmux execution results

```
node01:~ # shmux -c false -<$SHMX
```

```
 shmux! Child for node01 exited with status 1
>> Resuming... (Will pause on error)
 shmux! Child for node02 exited with status 1
 shmux! Child for node03 exited with status 1

3 targets processed in 10 seconds.
Summary: 3 errors
Error    : node01 node02 node03
```

More information about **shmux** is available at:

http://web.taranis.org/shmux/

watch - monitoring commands' output messages

watch is a small nifty tool (available on Linux) that can help you monitor changes and progress while executing repetitive commands. The **watch** command allows you to run a program periodically (similar to shell **while** loops), and continuously display or even sort the output and indicate (in reverse) where the changes are located.

An example of using the watch command is:

```
watch -n1 -d cat /proc/interrupts
```

This allows you to monitor the interrupts on your system (per CPU), and gives you an instant view of how well interrupts are balanced on a SMP system. If you're planning to bind certain hardware to specific CPUs, this helps you find and correct SMP affinity problems.

Another example allows us to follow progress in the RDAC driver:

```
watch -n1 -d cat /proc/mpp/DS4500_1/virtualLun0
```

5.3.2 System resource monitoring tools (Linux)

This section describes the Linux system resource monitoring tools.

dstat - versatile resource statistics tool

Dstat is a Linux replacement for tools like **vmstat**, **iostat** or **ifstat**. It gives you the possibility to show only the counters you require, using colors and units to avoid misinterpretation. It is modular, so it allows you to add your own counters in addition to the available set of modules.

The **dstat** tool comes with modules for monitoring CPU, memory, paging, disk, network, process, lock, load, process, interrupts per IRQ, shared memory, ipc, tcp, udp and many more counters.

In addition, **dstat** also has GPFS related modules that integrate with **mmpmon** and show you the GPFS I/O throughput, as well as the file system operation counters. It also has experimental modules for the RDAC, Qlogic and Emulex HBA driver, as well as several Samba modules.

The following Example 5-15 uses **dstat** to display the cpu (total), disk (sdc), gpfs io, network (bond0) and the five seconds load average.

Example 5-15 Dstat output relating CPU, system load, SCSI layer and GPFS throughput

```
node01:~ # dstat -c -d -D sdc -M gpfs -n -N bond0 -l 5
----total-cpu-usage---- --disk/sdc- --gpfs-i/o- -net/bond0- ---load-avg---
usr sys idl wai hiq siq|_read write|_read write|_recv _send|_1m_ _5m_ 15m_
  0   0 100   0   0   0|  21k   53k|   0   5.8B|   0     0 |  43   27   12
  1   6   0  93   0   0|   0    72M|   0    72M|  95B   88B|  43   28   12
  1   6   0  93   0   0|   0    71M|   0    71M| 274B   88B|  43   28   12
  1   6   0  93   0   0|   0    72M|   0    72M|  70B   88B|  43   28   12
  1   6   0  93   0   0|   0    71M|   0    71M|  70B   88B|  43   28   12
```

Example 5-16 shows **dstat** using the disk (sdc), GPFS operations, system and interrupts (network device, 101 and HBA device, 105):

Example 5-16 Dstat output relating SCSI layer, GPFS calls, resources and interrupts

```
node01:~ # dstat -d -D sdc -M gpfsop -yi -I 101,105 5
--disk/sdc- -------gpfs-operations------- ---system-- -interrupts
_read write|open clos read writ rdir inod|_int_ _csw_|_101_ _105_
  21k   94k|  0    0    0    0    0    0 |  28    22 |  26     0
   0    72M|  0    0    0   71    0    0 | 160  1911 |   7   146
   0    72M|  0    0    0   72    0    0 | 158  1948 |   8   146
   0    71M|  0    0    0   71    0    0 | 157  1854 |   8   144
   0    70M|  0    0    0   70    0    0 | 160  1897 |   8   142
   0    72M|  0    0    0   72    0    0 | 161  1993 |   7   147
```

You can see from the **dstat** command output that the GPFS blocksize used is effectively 1MB (each write operation maps to 1MB), and the HBA is using 512kB block sizes (each write causes roughly twice the number of interrupts).

These examples make clear that being able to see the isolated numbers next to each other make some relations more obvious and therefore helps you analyze problems faster.

However, if **dstat** does not provide all counters you may want, you can add your own module in minutes (after you studied the provided examples). Also, sharing

your modules with others will provide everyone with a more powerful tool. **Dstat** and the modules are written in Python.

More information about Dstat can be found at the Dstat Web site:

http://dag.wieers.com/home-made/dstat/

ifstat - Interface statistics

ifstat is a pretty straightforward tool, showing vmstat-like statistics for each interface. It helps finding bonding-related problems (as you will be able to relate the 'virtual' bonding interface with the physical interfaces.

The **ifstat** tool is different between SUSE SLES and Red Hat EL. Both tools provide the same kind of information using different command line options.

Example 5-17 shows the usage for the iproute **ifstat** tool on a SUSE SLES9 system:

Example 5-17 Sample ifstat output

```
node01:~ # ifstat -t 5
#21019.1804289383 sampling_interval=1 time_const=60
Interface        RX Pkts/Rate    TX Pkts/Rate    RX Data/Rate    TX Data/Rate
                 RX Errs/Drop    TX Errs/Drop    RX Over/Rate    TX Coll/Rate
lo                     0 0             0 0             0 0             0 0
                       0 0             0 0             0 0             0 0
eth0                  17 2            15 1          1360 231        4208 340
                       0 0             0 0             0 0             0 0
eth2                   0 0             0 0             0 22            0 8
                       0 0             0 0             0 0             0 0
eth3                   0 0             0 0             0 27            0 8
                       0 0             0 0             0 0             0 0
bond0                  0 0             0 0             0 49            0 16
                       0 0             0 0             0 0             0 0
```

mmpmon - GPFS performance monitoring

GPFS V2.3 ships with a new tool, mmpmon, that allows you to see a few internal GPFS performance counters. The "GPFS Administration and Programming Reference (bl1adm10)" explains how the command operates.

With mmpmon you can request:

- Per node I/O statistics (current node) - **io_s**
  - Total number of bytes read (disk/cache)
  - Total number of bytes written (disk/cache)

  - Number of open()/create() gpfs calls
  - Number of application read requests
  - Number of application write requests
  - Number of readdir() gpfs calls
  - Number of inode updates to disk
- Per filesystem I/O statistics - **fs_io_s**
  - Same counters as per node (see above)
- Size and latency ranged histograms - **rhist**

nfsstat - NFS performance monitoring tool

nfsstat gives you access to NFS system calls and rpc calls, for both the NFS server and NFS clients (on the system).

Here's the output for the NFS server after the counters have been reset and there's no activity:

Example 5-18 Sample iostat output

```
node01:~ # nfsstat -s
Server rpc stats:
calls      badcalls   badauth    badclnt    xdrcall
0          0          0          0          0
Server nfs v2:
null       getattr    setattr    root       lookup     readlink
0       0% 0       0% 0       0% 0       0% 0       0% 0       0%
read       wrcache    write      create     remove     rename
0       0% 0       0% 0       0% 0       0% 0       0% 0       0%
link       symlink    mkdir      rmdir      readdir    fsstat
0       0% 0       0% 0       0% 0       0% 0       0% 0       0%

Server nfs v3:
null       getattr    setattr    lookup     access     readlink
0       0% 0       0% 0       0% 0       0% 0       0% 0       0%
read       write      create     mkdir      symlink    mknod
0       0% 0       0% 0       0% 0       0% 0       0% 0       0%
remove     rmdir      rename     link       readdir    readdirplus
0       0% 0       0% 0       0% 0       0% 0       0% 0       0%
fsstat     fsinfo     pathconf   commit
0       0% 0       0% 0       0% 0       0%
```

If you're using nfsstat to monitor a used NFS server, you can use the watch command to follow progress in this manner:

```
watch -n1 -d nfsstat -s
```

This provides you with updates every second and indicates which numbers are changing. Since the nfsstat output is rather complex to look at and track changes, the -d (--differences) flag is particularly helpful.

5.3.3 Load generating tools

There are several load generating tools available, however none of these tools is able to mimic the particular load a real production environment (with different vendor devices) performs on the target system.

Most of these tools will create the maximum load possible and therefore strain the system much harder than it was designed for. Normally a system is designed only for a fraction of the estimated target workload, and will therefore behave much more differently under a maximum stress. In addition, usually these tools are not able to simulate the complex load generated by a real environment.

A proper tool to test a digital media infrastructure should allow to provide a sustainable (isochronous) load using multiple threads, orchestrated over multiple systems targeting multiple nodes in a cluster using file access patterns that mimics NLEs, play-out stations, ingests and specific vendor devices. Not a maximum stress. However, no such (free) tools exist at the moment.

dd

dd (device to device copy) is a basic UNIX tool. Although not originally intended for use as performance test and load generating tool, it has become the most used command for these purposes. It is an extremely versatile and simple to use tool, but the results you may get using this tool do not necessary reflect the real system behavior.

As it is a very low level command, capable of operating with all types of devices (block, character, special, files, etc.), it can be used for basic data transfer testing. You have to keep in mind that **dd** is not aware of the devices' characteristics, therefore the false results it may produce.

As an option you can use **dd_rescue**. For details on **dd_rescue**, see:

http://www.garloff.de/kurt/linux/ddrescue/

gpfsperf - GPFS performance data collection tool

gpfsperf is a tool that is distributed with GPFS as source code (see the samples directory /usr/lpp/mmfs/samples/perf). Check the README that comes with

gpfsperf carefully, as **gpfsperf** is not officially supported by IBM and provided as-is.

The gpfsperf tool is included as an example of how to use the gpfs_fcntl hints, and is useful to measure the GPFS file system performance for several common file access patterns. (sequential, strided and random)

To use it, you first have to compile it on the target system. For this, you need to install the development tools (like a compiler and development libraries). If everything is installed, do:

```
make -C /usr/lpp/mmfs/samples/perf clean all
```

On SLES9 ppc64 or x86_64 you may need also to use special options:

```
make -C /usr/lpp/mmfs/samples/perf clean all OTHERINCL=-m64 OTHERLIB=-m64
```

Then install with the following command:

```
install -Dp -m0755 /usr/lpp/mmfs/samples/perf/gpfsperf
/usr/local/bin/gpfsperf
```

The README included with **gpfsperf** also explains how to use **gpfsperf**. The examples included in the README however are aimed at the **gpfsperf-mpi** program which requires MPI (Message Passing Interface) and an mpi compiler to compile it. We were not able to compile gpfsperf-mpi against mpich-devel.

The **gpfsperf-mpi** tool allows you to run **gpfsperf** orchestrated on multiple nodes at the same time.

The syntax of **gpfsperf** is:

```
gpfsperf operation pattern filename [options]
```

Where:

- Operation is either “create”, “read”, “write” or “uncache”
- Pattern is either “rand”, “randhint”, “strided” or “seq”
- Options are one ore more of the following:
  - -r recsize

    Record size (defaults to file system blocksize)
  - -n bytes

    Number of bytes to transfer (defaults to file size), can be followed by a unit, e.g., 64m, 16g or 1024r (which is 1024 times the record size)
  - -th threads

    Number of threads per process (defaults to 1)

Example 5-19 shows how you can run **gpfsperf** to create a 4 GB file, using 1 MB blocksizes and 16 threads:

Example 5-19 gpfsperf sequential create test

```
node01:~ # gpfsperf create seq /gpfs/gpfstest -n 4g -r 1m -th 16 -fsync
gpfsperf create seq /gpfs/gpfsperf-test
  recSize 1M nBytes 4G fileSize 4G
  nProcesses 1 nThreadsPerProcess 16
  file cache flushed before test
  not using data shipping
  not using direct I/O
  not using shared memory buffer
  not releasing byte-range token after open
  fsync at end of test
    Data rate was 74694.64 Kbytes/sec, thread utilization 0.901
```

Now that the file exists, we can do a similar sequential read test (Example 5-20):

Example 5-20 gpfsperf sequential read test

```
node01:~ # gpfsperf read seq /gpfs/gpfsperf-test -r 1m -th 16
gpfsperf read seq /gpfs/gpfsperf-test
  recSize 1M nBytes 4G fileSize 4G
  nProcesses 1 nThreadsPerProcess 16
  file cache flushed before test
  not using data shipping
  not using direct I/O
  not using shared memory buffer
  not releasing byte-range token after open
    Data rate was 179990.84 Kbytes/sec, thread utilization 0.989
```

Make sure there is no cached data, and then run a random read test, as shown in Example 5-21:

Example 5-21 gpfsperf random create test

```
node01:~ # gpfsperf read uncache /gpfs/gpfsperf-test
uncache /gpfs/gpfsperf-test
node01:~ # gpfsperf read rand /gpfs/gpfsperf-test -r 1m -th 16
gpfsperf read rand /gpfs/gpfsperf-test
  recSize 1M nBytes 4G fileSize 4G
  nProcesses 1 nThreadsPerProcess 16
  file cache flushed before test
  not using data shipping
  not using direct I/O
  not using shared memory buffer
  not releasing byte-range token after open
```

```
Data rate was 204784.35 Kbytes/sec, thread utilization 0.984
```

5.4 Client application considerations

It's important to understand that much effort and money can be spend trying to create a highly available or even fault tolerant infrastructure that is able to recover from any failure in a short time. But none of the technically available solutions can compete against a solution with proper client side buffering, client side load balancing and client side failover mechanisms.

If you have the ability to influence client application design or development, we provide you with some considerations for a digital media environment.

5.4.1 Client side buffering and failover mechanism

With proper client side buffering the application can overcome hiccups in any of the infrastructure's components, and, if necessary, switch over to a secondary server. In a digital media environment, specially in playback mode (broadcasting), for fulfilling the isochronous requirement, it is important to have a large buffer on the client side, and also the ability to feed this buffer (if necessary) from multiple sources. Only the client application knows when a feed is not meeting its requirements (for whatever reason) and can make up for this using a redundant path without noticeable interruption.

5.4.2 Client side load balancing

On the server side none of the existing load balancing techniques offers true load balancing (only a Layer 7 based implementation could technically offer transparent load balancing, and such implementation does not exists at this point for CIFS or NFS). The problem here is hiding the protocol's complexity from the client application so that the client application does not need to know or care. Still the easiest implementation for proper load balancing is to involve the client in the process either by feedback from a daemon running on the cluster nodes, or by IPC™ between other client applications.

5.4.3 Block size considerations

GPFS allows you to choose between different block sizes, and we've already seen that choosing the optimal blocksize is not straight forward. The GPFS block size depends on the disk block size, the LUN stripe size, the application block size, and even on the typical file size. Of all these variables only the application block size might be configurable in your environment. If you can afford, it would

be useful to allow an application configurable block size, and perform tests with several block sizes in your environment.

To avoid overhead it is advisable to use the same block size in the client application as the GPFS block size. If this is not possible, we recommend to choose multiples or sub-multiples of the GPFS block size.

If the application is directly access the GPFS file system (no NFS or CIFS), the application can query the file system for the block size using the POSIX statvfs() system call and looking at the f_bsize value (Linux).

5.4.4 Mixed-media processing and file access patterns

We have learned that GPFS can drastically improve the I/O performance to the storage backend if it involves sequential access. For (true) random I/O, GPFS will not be able to use read-ahead or write-behind techniques and can only rely on the striping for improved performance.

Analyzing applications' I/O pattern

It is also important to look at the clients' file access patterns before implementing your solution and during performance tests to understand and streamline total throughput. Multimedia files are often containers that consist of multiple streams (e.g., video streams with multiple audio channels). When using these files from a client application, the performance can be improved by opening the file multiple times and reading each stream separately. For GPFS this will transform a single random file access pattern into several strided (or even sequential) reads.

There are several tools to look at file access patterns of applications:

- **`strace`** and **`ltrace`** on Linux
- **`truss`** on AIX
- **`filemon`** for Windows

And, there are tools to look at file access patterns in the network:

- **`ethereal`** for Linux, AIX and Windows
- **`tcpdump`** for Linux

Using these tools one can provide file access logs on several layers, for example:

1. Using **`filemon`** on Windows to analyze client's file access patterns
2. Using **`ethereal`** to analyze the CIFS (Windows Network) file access patterns
3. Using **`tcpdump`** or **`ethereal`** to analyze the file access patterns on Linux
4. Using **`strace`** on Linux to analyze Samba's network and disk access

5. Using **mmfsadm** and **mmpmon** to correlate the access patterns

Comparing information gathered by these tools gives an insight view of what is happening, helps finding possible performance problems, and gives potential to improve the client applications' file access patterns. This also allows you to understand and correlate the behavior concerning block sizes, file operations and latency.

Alternatives

In case it is not possible to open a file multiple times to access these different streams (because used libraries do not allow this level of control) it might still be possible to influence GPFS's read-ahead behavior by using big enough blocksizes to access the multimedia files so that subsequent access of other streams can be fetched from the pagepool or (again if the application has direct access to the GPFS filesystem) to use GPFS hinting techniques to influence GPFS' decisions.

5.5 Performance tuning considerations

This section provides a list of places to check for performance tuning in a GPFS/DM environment.

Performance tuning

1. Look at the sysctl entries.
2. Check pagepool settings (optimal: 256 MB, max: 2 GB, default: 20 MB).
3. Set MaxMBpS (optimal: 400 MB, default: 150 MB).
4. Malloc pool settings.
5. maxFilesToCache (default 200 inodes to cache).
6. Change priority for mmfsd (default: 40) mmchconfig priority=42.
7. For AIX, set the rpoolsize/spoolsize (optimal: 16MB -> 16777216).
8. Disable hyperthreading as Samba uses a single working process.
9. Bind Samba and GPFS to a separate CPU.
   - for pid in $(pidof smbd); do taskset -p 0x01 $pid; done
   - for pid in $(pidof mmfsd); do taskset -p 0x02 $pid; done
10. Increase the negotiated buffer size on both Windows clients as well as on Linux (max xmit), set to 64 KB

    http://article.gmane.org/gmane.network.samba.internals/10807

    May require tweaking to customer case.

11. Enable O_DIRECT in Samba (disable caching).
12. Mount SMB shares on Linux with:

```
-o guest,sockopt=TCP_NODELAY,IPTOS_LOWDELAY,SO_SNDBUF=131072,SO_RCVBUF=65536
```

13. Disable sendfile when having problems.

Bottleneck tests

1. Check Direct-IO compatibility with GPFS support team.
2. Check that we use 32 Kb or 64 Kb blocks in SMB.
3. Check the registry settings in Windows.
4. Do a test with only *one* stream per samba session.
5. Check for dead time (unjustified I/O wait time).
6. Perform tests with NFS between Linux systems.
7. Perform a test with many workstations (can be VMware instances).
8. Perform a test from the loadbalancer.
9. Check if you can increase blocksize with smbmount (Linux SMB client).
10. Set oprofile=2 for profiling.
11. Enable jumbo frames (less interrupts, more CPU).

5.6 List of GPFS related files

This section provides a table containing the files of interest for a system administrator in a GPFS/DM/Linux/AIX environment.

Table 5-6 List of files

Filename	Description
Common AIX and Linux	
/etc/exports	Defines the exported NFS filesystems and the NFS options per filesystem
/etc/fstab	Defines the filesystems that are locally mounted and the options per filesystem
/var/mmfs/etc/mmfsdown.scr	Script that is executed when stopping GPFS
/var/mmfs/etc/mmfsup.scr	Script that is executed when starting GPFS
/var/mmfs/gen/mmfsNodeData	Generated file containing the GPFS cluster node information **Do not edit!**

Filename	Description
/var/mmfs/gen/mmsdrfs	Generated file containing the complete GPFS cluster information and locally attached disk definitions. **Do not edit!**
Common SUSE Linux Enterprise Server and Red Hat Enterprise Linux	
/etc/modprobe.conf	Defines a list of modules and options per module
/etc/mpp.conf	Lists the options of the RDAC driver
/etc/ntp.conf	Main NTP configuration file
/etc/ntp/ntpservers	Lists the NTP servers used for initial synchronization
/etc/sysctl.conf	Lists all the permanent kernel related tuning parameters
/var/mpp/devicemapping	Maps the known storage devices into a list for the RDAC driver
SUSE Linux Enterprise Server only	
Red Hat Enterprise Linux only	
/etc/samba/lmhosts	Maps hosts statically with IP addresses (instead of using Local Master browser)
/etc/samba/smb.conf	Main configuration file for Samba, contains the options, exported Samba shares and ACLs
/etc/samba/smbpasswd	Lists the user passwords for authenticating to Samba
/etc/samba/smbusers	Lists the users that can authenticate to Samba

A

Basic infrastructure configuration additions

This appendix provides the following information:

- Automatic network installation using SLES 9 Installation Server
- Configuring tftpd
- Configuring dhcpd
- SLES9 Installation Server Configuration with YaST using FTP
- Configuration of secure shell
- Setting up storage software
- Installing and configuring a distributed shell
- SLES9 Installation on an IBM eServer BladeCenter JS20
- Sample storage profile file for DS4500

A.1 Automatic network installation using SLES 9 Installation Server

To be able to boot the client from the network, you need five things:

1. Client needs a network card capable to boot from the network.
2. Service that answers queries from the network card.
   - "PXE" is used for network boot on Intel-baed computers.
   - "elilo" is used for network boot for Itanium® Processor Family computers
3. tftpd needs to be configured (see A.2, "Configuring tftpd" on page 212).
4. An installation Server (see A.4, "SLES9 Installation Server Configuration with YaST using FTP" on page 215)
5. dhcpd needs to be configured ().

In the following we will describe how to set up an installation server using SLES9 on an Intel-based platform and the steps necessary for booting Intel- or POWER based systems.

A.2 Configuring tftpd

This section describes how to configure tftpd for booting Intel-based computers (PXE).

Configuring tftpd for booting Intel-based computers (PXE)

You need to copy the following files to the /tftpboot directory:

- pxelinux.0 located in /usr/share/syslinux/pxelinux.0 part of the syslinux package. You may have to install the syslinux package by using YaST or rpm.
- linux and initrd located in the /boot/loader directory of the first installation CD. of the x86 collection

Finally you have to create at least a configuration file called "default" in the /tftpboot/pxelinux.cfg directory as shown in the following Example A-1:

Example: A-1 Default PXE config file located in /tftpboot/pxelinux.cfg

```
default linux
label linux
  kernel linux
  append initrd=initrd ramdisk_size=65536 install=slp:
```

```
implicit        0
display         message
prompt          1
timeout         200
notice          2
```

It is also possible to use a dedicated configuration for specific IP addresses. If you like to create a configuration file, i.e., for IP address 192.168.100.212, you need to determine its name with the command “gethostip 192.168.100.212”. In this case the name of the configuration file will be: C0A864D4

A.2.1 Configuring tftpd for booting POWER-based computers

You need to copy the following files to the /tftpboot directory:

- Image.prep.initrd located in the /boot directory of the first installation CD of the PPC distribution.
- The /install file from the SLES 9 CD #1 to the /tftpboot directory.

  - **Note:** Make sure the files in /tftpboot have the correct owner and group IDs assigned. Since the tftpd process usually does not run as the root user, check the tftp line in /etc/inetd.conf. In our example, “-u tftpd” is specified. The user and group IDs of the files in /tftpboot must match the information in the system permission database for the user “tftpd”. To match the example, run:

    chown tftpd:tftpd /tftpboot/install

- To create a customized kernel for the unattended installation, copy the install kernel into the /tftpboot directory and issue the command “mkzimage_cmdline” as shown in Example A-2. This will avoid running through the SMS procedure, which means to just boot through TFTP, and the kernel will start linuxrc with the right arguments. A more detailed description can be found in the /ppc/netboot directory.

Example: A-2 “Maintain” the kernel for unattended installation

```
mkzimage_cmdline -a 1 -c -s "insmod=bcm5700
install=ftp://192.168.100.242/install-tree/sles9_ppc/
autoyast=ftp://192.168.100.242/install-tree/sles9_ppc/yastcfg/ netdevice=eth1"
install-ppc-sp1
```

Tip 1: An entry in the arp cache is needed on the server which hosts the tftp service. Remove the static entry after install is complete.

Tip 2: Use atftp instead of tftp because it offers more logging possibilities and is much more robust. Make sure that the service will run as a standalone daemon instead of being under control of inetd or xinetd.

Tip 3: Use http, instead of ftp or NFS, for three reasons:

- Debugging is easier as http logs more
- Routing is possible (firewall, proxy)
- ISO-images are usable

A.3 Configuring dhcpd

You need to configure a DHCP server for being able to automatically boot over the network and assign the correct IP address and boot image to the client. So configure the /etc/dhcpd.conf file like shown in Example A-3. Adjust the filename, host name, hardware ethernet (MAC) address to match your environment. Note that the sections for the hardware platform defer.

Example: A-3 example for /etc/dhcpd.conf file

```
# dhcpd.conf

# option definition common to all supported networks ...
allow bootp;
allow booting;
always-reply-rfc1048    true;
deny unknown-clients;
not authoritative;

default-lease-time      600;
max-lease-time          720;
ddns-update-style       none;
log-facility local7;

subnet 192.168.100.0 netmask 255.255.255.0 {
  group {
    next-server 192.168.100.241;
    host js20n11 {
#       hardware ethernet 00:0d:60:1e:ca:00;
      hardware ethernet 00:0d:60:1e:bf:99;
      fixed-address 192.168.100.211;
      filename "z.Image.prep.initrd";
    }
```

```
    host js2On12 {
#      hardware ethernet 00:0d:60:1e:cd:48;
      hardware ethernet 00:0d:60:1e:cd:49;
      fixed-address 192.168.100.212;
      filename "z.Image.prep.initrd";
    }
    host js2On13 {
#      hardware ethernet 00:0d:60:1e:bf:84;
      hardware ethernet 00:0d:60:1e:bf:85;
      fixed-address 192.168.100.213;
      filename "z.Image.prep.initrd";
    }
    host js2On14 {
#      hardware ethernet 00:0d:60:1e:ca:00;
      hardware ethernet 00:0d:60:1e:ca:01;
      fixed-address 192.168.100.214;
      filename "z.Image.prep.initrd";
    }
    host lnx03 {
      hardware ethernet 00:00:00:00:00:00;
      hardware ethernet 00:00:00:00:00:00;
      fixed-address 192.168.100.243;
      filename "pxelinux.0";
    }
  }
}
# end of dhcpd.conf
```

Note: You must insert a permanent ARP entry for the MAC address of every host to avoid that the boot process stops.Remove the static entry after installation is complete.

A.4 SLES9 Installation Server Configuration with YaST using FTP

If you need a more detailed description or an alternative to the here described FTP method, please check the SLES manual located on SLES9 CD1under /docu.

- Start yast from a command line and choose “Misc” followed by “Installation Server”.
- Select the server type. Choose FTP and enter the path where the source packages will be located (see Figure A-1 on page 216).

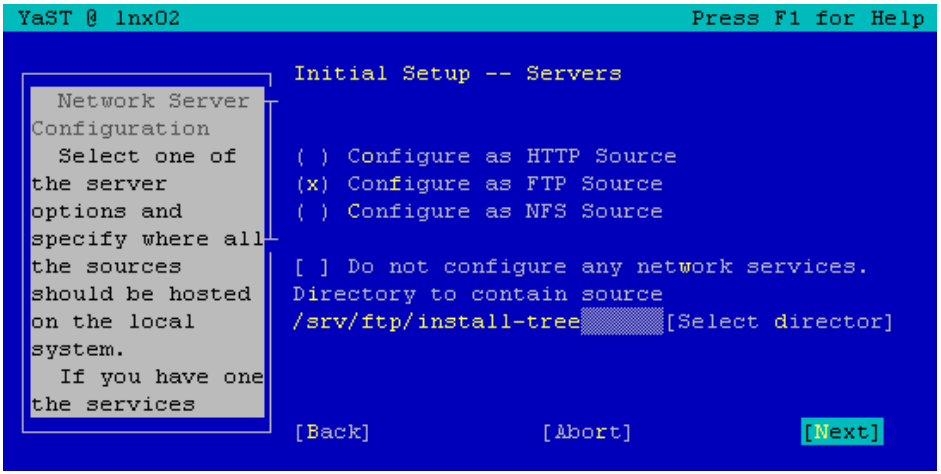

Figure A-1 YaST Installation Server - selecting the Server Type

- Choose the FTP root directory and optionally Directory Alias as desired (see Figure A-2).

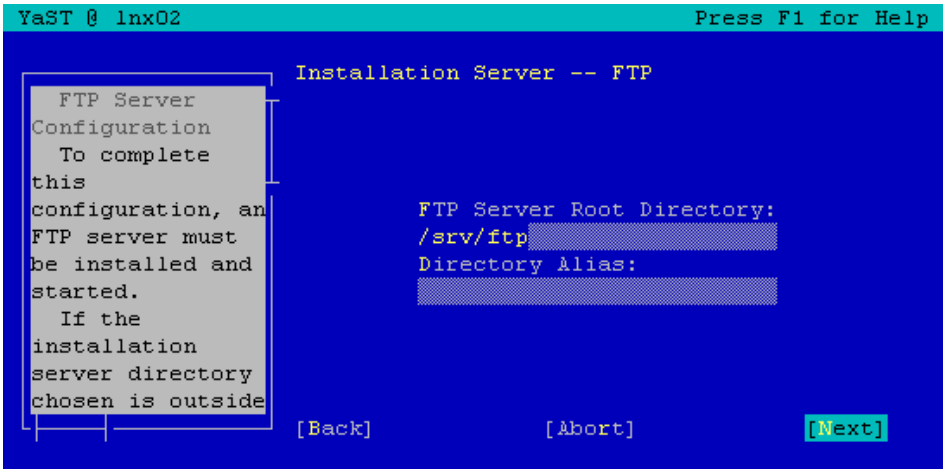

Figure A-2 YaST Installation Server - FTP configuration

- YaST will now copying the actual SUSE Linux CDs. This is the most time consuming step. If the installation server should provide more than one package or version, start YaST module Installation Server module and select “Configure”. Add further versions as desired (see Figure A-3 on page 217).

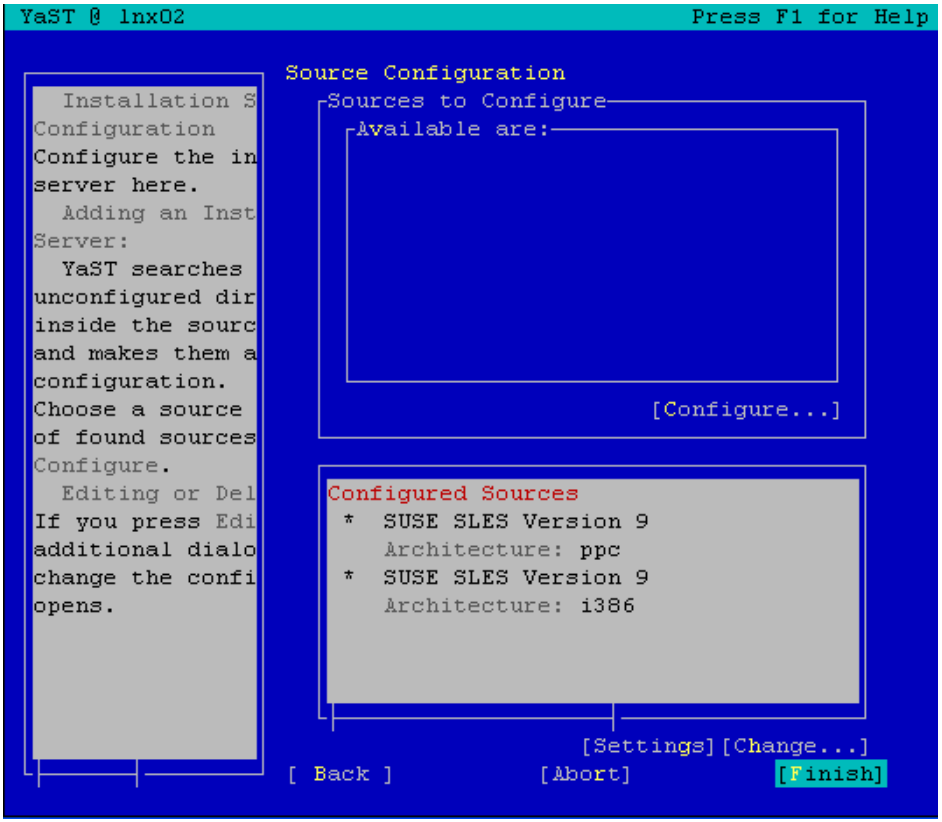

Figure A-3 YaST Installation Server - Overview of Installation Sources

- Create the AutoYaST control file. We did a manual installation of one machine first and saved the profile of this machine by using AutoYaST. With slightly changes you can reuse this for other JS20 hosts as input file for the unattended installation. Transfer this file to the /install-tree/sles9/yastcfg directory on the install server.

Note: Use eth1 for the installation process since eth0 is used for SOL.

- Finally patch kernel options on the install server. This is done using a utility that comes with SLES 9 called mkzimage_cmdline. This utility is located in the /ppc/netboot directory of SLES 9 CD #1. Based on the install server setup we've done, the mkzimage_cmdline we would invoke would be:

```
mkzimage_cmdline –s
“autoyast=ftp://192.168.100.241/sles9-ppc-base/yastcfg/install=ftp://192.16
8.100.241/sles9-ppc-base netdevice=eth1” –a 1 install
```

Instead of using the BladeCenter® Management Module GUI to change the boot sequence, the open firmware implementation offers to change it at the boot prompt (or remotely by using SOL).

A.5 Configuration of secure shell

AIX only

Secure shell packages can be downloaded from various sites, but for specific implementations, we recommend the use of the ones directed from IBM. To install Open SSH packages you also need the Open SSL, both available from (directly or through additional Web links):

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=aixtbx

Important: To access these files, you need to register for an IBM ID. Plan carefully, ad registration may take up to 24 hours!

The secure shell package installation is performed only on AIX (which does not come with SSH by default). The ssh configuration part is similar on AIX and Linux.

- Obtain the OpenSSL and OpenSSH packages

Attention: Check your clustering software for special OpenSSL and OpenSSH requirements. For example, with GPFS always check the latest GPFS FAQ list at:

http://publib.boulder.ibm.com/infocenter/clresctr/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfsclustersfaq.htmlh

On Linux, the version of OpenSSL included with your Linux distribution should be used.

On AIX 5L™, OpenSSL 0.9.7-2 or later, as distributed by IBM, is required. Other publicly available versions of OpenSSL may not work. The required version of OpenSSL is included in the AIX Toolbox for Linux Applications.

- Depending on the packages you have, you may use either installp or rpm to install the packages.

Restriction: Although the SSL version is the same for both AIX V5.2 and V5.3, the Open SSH package may be compiled different (different libraries), therefore make sure you download and install the correct version.

- Install and verify the packages:

Example: A-4 OpenSSL/OpenSSH packages installed (AIX 5.3 ML01)

```
[p630n02][/.ssh]> lslpp -L |grep open
  openssh.base.client     3.8.0.5302    C     F    Open Secure Shell Commands
  openssh.base.server     3.8.0.5302    C     F    Open Secure Shell Server
  openssh.license         3.8.0.5302    C     F    Open Secure Shell License
  openssh.man.en_US       3.8.0.5302    C     F    Open Secure Shell
  openssl                   0.9.7d-2    C     R    Secure Sockets Layer and
  openssl-devel             0.9.7d-2    C     R    Secure Sockets Layer and
  openssl-doc               0.9.7d-2    C     R    OpenSSL miscellaneous files
[p630n02][/.ssh]>
```

And for AIX 5.2 ML04:

Example: A-5 OpenSSL and OpenSSH packages on AIX 5.2ML04

```
[p630n01][/]> lslpp -L |grep open
  openssh.base.client     3.8.0.5202    C     F    Open Secure Shell Commands
  openssh.base.server     3.8.0.5202    C     F    Open Secure Shell Server
  openssh.license         3.8.0.5202    C     F    Open Secure Shell License
  openssh.man.en_US       3.8.0.5202    C     F    Open Secure Shell
  openssl                   0.9.7d-2    C     R    Secure Sockets Layer and
  openssl-devel             0.9.7d-2    C     R    Secure Sockets Layer and
  openssl-doc               0.9.7d-2    C     R    OpenSSL miscellaneous files
[p630n01][/]>
```

- Configure and start the ssh server (sshd) on all nodes in your cluster. When you install the packages, the sshd will be automatically added to the AIX System Resource Controller and started. During the installation process, the **server** keys will be generated automatically and stored in /etc/ssh directory, as shown in the following example.

Example: A-6 SSH server (daemon) and configuration files

```
[p630n01][/]> lssrc -a|grep ssh
 sshd             ssh              13946        active
[p630n01][/etc/ssh]> ls -all
total 272
drwxr-xr-x   2 root     system          512 Feb 25 20:31 .
drwxr-xr-x  25 bin      bin            7168 Jun 05 14:43 ..
-rw-------   1 root     system        88039 Feb 25 20:24 moduli
-rw-r--r--   1 root     system         1310 Mar 30 13:57 ssh_config
-rw-------   1 root     system          672 Feb 25 20:31 ssh_host_dsa_key
-rw-r--r--   1 root     system          590 Feb 25 20:31 ssh_host_dsa_key.pub
-rw-------   1 root     system          515 Feb 25 20:31 ssh_host_key
-rw-r--r--   1 root     system          319 Feb 25 20:31 ssh_host_key.pub
-rw-------   1 root     system          887 Feb 25 20:31 ssh_host_rsa_key
-rw-r--r--   1 root     system          210 Feb 25 20:31 ssh_host_rsa_key.pub
-rw-r--r--   1 root     system         1418 Feb 25 20:24 ssh_prng_cmds
-rw-r--r--   1 root     system         2425 Feb 25 20:31 sshd_config
```

```
[p630n01][/etc/ssh]>
```

AIX and Linux

Collecting server keys in a “known_host” file

- Check if ssh is working: from each node, connect to the other nodes using “ssh” command.

Example: A-7 Testing ssh

```
[p630n01][/etc/ssh]> ssh root@p630n02
The authenticity of host 'p630n02 (192.168.100.32)' can't be established.
 . key fingerprint is RSA.
 Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.100.32' (RSA) to the list of known hosts.
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 5.2!                                                *
*                                                                             *
*******************************************************************************
[p630n02][/]>
```

- Each node in you are connecting to sends it’s public key which will be stored into the ~/.ssh/known_hosts file on the node that has initiated the ssh connection (client). There is one known_hosts file per user (identity). This file stores the public keys of all servers (sshd) that you have connected to. The server public key is used to encrypt the traffic between the client (ssh) and the server you connect to.

Attention: If you have multi-homed systems (i.e., multiple IP addressed/adapters on each node), you need to connect to the other systems using all available IP addresses/labels you plan to use for remote cluster commands.

- On one node (you can designate it as administrative node), connect to ALL nodes using ALL interfaces/IP addresses, including the LOCAL ones. Check the ‘/.ssh/known_hosts file to see if you have one entry for each IP you have connected to.

Generating “authorized_keys” file

An authorized_keys file contains all public keys of the remote client identities (user@hostname) that are allowed to connect and/or execute remote commands under a certain identity on a specific node.

For GPFS, all commands must be executed with “root” authority.

- As user “root”, generate the private/public key pair (RSA) for EACH node in the cluster:

Example: A-8 Key generation for ssh

```
[p630n01][/]> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (//.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in //.ssh/id_rsa.
Your public key has been saved in //.ssh/id_rsa.pub.
The key fingerprint is:
4b:0b:f0:a4:6a:9e:3c:9e:93:d7:b6:01:f5:2a:be:c6 root@p630n01
[p630n01][/.ssh]> ls -al
total 120
drwx------   2 root     system          512 Jun 05 17:39 .
drwxr-xr-x  32 root     system         1536 Jun 05 11:18 ..
-rw-------   1 root     system          883 Mar 30 13:42 id_rsa
-rw-r--r--   1 root     system          222 Mar 30 13:42 id_rsa.pub
-rw-r--r--   1 root     system         4347 Jun 05 17:30 known_hosts
[p630n01][/.ssh]>
```

- To allow root access to all the nodes in the cluster, you have to create a file named ~/.ssh/authorized_keys on each node in the cluster, containing the RSA public keys of all the nodes in the cluster.

Important: It is essential that you generate the client side keys **without** a pass phrase for the private key. If you fail to do so, you will be asked for the pass phrase each time you want to execute a remote command, therefore defeating the purpose of remote command execution without a password.

- On the same (administrative designated) node:
  - Collect the id_rsa.pub files from all nodes (using scp, ftp, or a diskette, if you do not trust your network) and concatenate them in a file named ~/.ssh/authorized_keys.
  - Make sure the permission for this file is 600
  - Distribute (copy) the local ~/.ssh/authorized_keys file to all nodes in the cluster (to ~/.ssh/authorized_keys).
  - Distribute also the ~/.ssh/known_hosts to all nodes in the cluster.

Note: Generating the ~/.ssh/known_hosts and ~/.ssh/authorized_keys on a node makes administrative work much easier. In this way you can maintain keys consistency.

Example: A-9 Sample ~/.ssh/authorized_keys file

```
[p63On01][/.ssh]> cat authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAsSC1ItuN/zNerPSS93rOMdwVit3JAGH4QXMxHbXZYgH/QMWIRBI
+CNv4yHIlkuvKaqaAEFzj+UTK1ACofz/xgcPiM6JKL9ItBNOzRzawEI8ltpMRcn2/+tIA/OKllaVsoe
yBgke2OnRjqr1zD/Om8ezwLJAmrqUyECt8LXAOLfO= root@p63On01
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAxjPtx5nzcQ6D9k18FNOmwOyPDkaEs2RzthpWmZtx7JgfzKPyvdu
1M2KbzidM8jfQnw6wUL2SwGk8wTwIabQVHJ9zRDDOVzss4933KF/ubIkA+KxOEIfJOan1UIdhogdzY+
V58tvXAxh7F42vVvqyFo3TRvmI7CwFWtWXTaJTF/M= root@p63On02
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAxplXMVAWeXwdxlp8f75gWIR+NzOIKjUwv86TPFQlwdpDPUAJpHV
5tBZWHWOW3X7nNZkp+BtuFOIlpPf9s2U/d+InMcu3h8UKABzlhmPhol6GOiY5rqeTVC7m2QxwXdWmfQ
nIzfDvzpmWczWC2MBFXZTyJK59S3KWBOrwUcI9Fdk= root@p63On03
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEA3bO4azxDmwHw7vEqBLwPH/GvgRmvBHABMV5BaToXedAH7LeJO41
pnxtuKwHJAdopXoVA/T4yqIICxtXxM37Zkeq+BHvR1q3phKLfgleUNZzDKunSrHnFEFaGKJHjTEDlnw
JEFCjEZuOwkeGS2y2WJkI1DfV2xyonSwFSumOEI6O= root@p63On04
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAvOllJlsbs4xmj5k6Zi82PijMb8KeSXNhDkjN14TXWjG5B/Nr2+b
PGEatrQk6yVcWGeRIFgQWdSiai2Akox5uTOJgiha9SfmKbwLjxIuiMd7gcTcModgbOWZP+GlF7rIWCB
tyJn7QbbinfxXy6ETwEweQlJH3KllOV+yQ5OFrn1c= root@p63On05
ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAz2ph+cIOOY/8EHL8j9n39FYxXSdoggQdQ824PyHiZGggvtGFBKr
ulRiLtR3P3SmcsTB7aGr+EV1LZupQ7LLm+5qLYOVg5McvcPSWMHudhw8/QCHSO9wgCruMldb9UG7Lny
DL3SJFN/cFzPr4cakF7Lo4+YXj3dilOCGpZURcOBO= root@p63On06
```

From each node, using all interfaces / IP labels, verify remote command execution without prompting for a password, by issuing: "**`ssh IP_label date`**".

A.6 Setting up storage software

We downloaded the latest version of the Storage Manager for the DS4000 family which is V9.12 at the time written. The package consists of the following software components for Linux:

- Runtime (SMruntime)
  This component contains the Java JRE for other software packages.
- Client (SMclient)
  This component is necessary if you intend to manage the storage subsystem from a host computer or a management station.
- Utility (SMutil)
  This component packaged with the SMclient package contains the hot_add and SMdevices utilities.

- RDAC (SMRDAC)
  This component provides multi path fail over/fail back capability.
- Agent (SMagent)
  This component is necessary if you intend to use the host-agent (in band) management method. It requires that you first install RDAC.

As we intend to use the out-of-band method only to configure the DS4000, we will only use the first four packages. On the management station we installed the SMruntime and the SMclient package to get the GUI and the CLI in place to manage the DS4000. On the other hosts, we just installed the SMRDAC and SMutil packages.

Storage Manager (GUI) and storage management agents

The Storage Manager offers different options of management software. We installed the client GUI on our management station.

A.7 Installing and configuring a distributed shell

To work more efficiently, you can use a tool, i.e., called shmux, which can be downloaded here (there are many others around as well):

http://web.taranis.org/shmux/#download

You can think of it as a distributed shell, as it is a program for executing the same command on many hosts in parallel.

http://linux01.gwdg.de/~pbleser/aboutme.php
http://ftp.gwdg.de/pub/linux/misc/suser-guru/rpm/9.3/RPMS/src/

The syntax and output is shown for the date command in Example A-10. The nodes where the specified command should be executed are specified in the file /tmp/nodes. You need to put the host names or IP addresses of the hosts in there, each in a separate line.

Example: A-10 syntax and result for shmux usage

```
shmux -c "date" -</tmp/nodes
p5n1lp1: Wed May 18 11:01:56 EDT 2005
p5n1lp2: Wed May 18 11:02:01 EDT 2005
p5n2lp1: Wed May 18 11:01:54 EDT 2005
p5n2lp2: Wed May 18 11:01:56 EDT 2005

4 targets processed in 1 second.
Summary: 4 successes
```

A.8 SLES9 Installation on an IBM eServer BladeCenter JS20

As an example we describe here how to set up SLES9 on an IBM @server BladeCenter JS20.

Note: The BladeCenter JS20 is for now only supported as GPFS NSD client.

White papers related to the BladeCenter can be found here:

http://www-1.ibm.com/servers/eserver/bladecenter/literature/wp_lit.html

Regarding the installation of SLES9 on BladeCenter JS20 follow the instruction described next for a manual installation. For a more detailed installation guide, and also for other installation options, check:

http://www-1.ibm.com/servers/eserver/bladecenter/literature/pdf/bctrjs20sles8install090704.pdf

- Connect from a host with a Web browser to the management module of the BladeCenter.
- Configure the chosen blades for Serial Over LAN (SOL) access. This step is required in order to get access to the system because the BladeCenter JS20 has no direct serial connection for a monitor, keyboard or mouse.
- Under Blade Tasks, select **Configuration**, then **Boot Sequence**. Select the server to be installed, change the boot sequence to boot from CD-ROM first and click **Save**.
- Open a telnet session to the management module of the BladeCenter and enter the userid/password, the default is USERID/PASSWORD (all upper case, with a zero).
- Increase the time out for a telnet session to the required time frame for installation, that is:

  ```
  telnetcfg -t 7200
  ```

- Connect to the required blade by entering the following command:

  ```
  env -T system:blade[#]
  ```

- Check that SOL is on:

  ```
  check sol -status on
  ```

- Get the console of your blade by using this command:

  ```
  console -o
  ```

  To terminate an SOL session, press ESC, then shift-9 (a left parenthesis).

The SOL session puffers up to 8 KB of data, so when the session is started, any buffered data will scroll past.

- Put the SLES9 CD#1 into the media tray's CD-ROM drive and give the target install blade sole access to the CD-ROM drive by pressing the CD button on the top/front of the blade. Reboot the blade.
- When the boot prompt of SLES shows up, you have two options to choose from. You can use a telnet or vnc session during SLES installation.Depending on your choice (text- or GUI-based), you need to enter:

```
boot:>install console=hvsi0
boot:>install vnc=1 vncpassword=password
```

- During YAST configuration choose your settings.
- Finally reboot the system (blade).

A.9 Sample storage profile file for DS4500

This section contains a listing of the DS4500 storage subsystem profile we used in our environment.

Example: A-11 Sample storage profile

```
PROFILE FOR STORAGE SUBSYSTEM: DS4500_1 (09.06.05 11:34:32)

SUMMARY------------------------------
   Number of controllers: 2
   Number of arrays: 8
   Total number of logical drives (includes an access logical drive): 9 of 2048
used
      Number of standard logical drives: 8
      Number of access logical drives: 1
      Number of flashcopy repositories: 0
   Number of flashcopy logical drives: 0
   Number of drives: 28
   Supported drive types: Fibre (28)
   Total hot spare drives: 0
      Standby: 0
      In use: 0
   Access logical drive: None mapped
   Default host type: LNXCL (Host type index 13)
   Current configuration
      Firmware version: 06.12.03.00
      NVSRAM version: N1742F900R912V06
   Pending configuration
      Staged firmware download supported?: Yes
      Firmware version: None
```

```
      NVSRAM version: None
      Transferred on: None
   NVSRAM configured for batteries?: Yes
   Start cache flushing at (in percentage): 80
   Stop cache flushing at (in percentage): 80
   Cache block size (in KB): 16
   Media scan frequency (in days): 30
   Failover alert delay (in minutes): 5
   Feature enable identifier: 353834383700000000000000040323AA0
   Storage Subsystem worldwide name (ID): 600A0B8000120F33000000004051A03D

CONTROLLERS------------------------------
   Number of controllers: 2

   Controller in Slot A
      Status: Online
      Current configuration
         Firmware version: 06.12.03.00
            Appware version: 06.12.03.00
            Bootware version: 06.10.04.00
         NVSRAM version: N1742F900R912V06
      Pending configuration
         Firmware version: None
            Appware version: None
            Bootware version: None
         NVSRAM version: None
         Transferred on: None
      Board ID: 5884
      Product ID: 1742-900
      Product revision: 0520
      Serial number: 1T34058487
      Date of manufacture: 10. Oktober 2003
      Cache/processor size (MB): 1024/128
      Date/Time: Thu Jun 09 11:33:55 EDT 2005
      Associated Logical Drives (* = Preferred Owner):
         GPFS_data1*, GPFS_data3*, GPFS_metadata1*, GPFS_metadata3*
      Ethernet port: 1
         MAC address: 00:a0:b8:12:0f:33
         Host name: fast9itso1
         Network configuration: Static
         IP address: 192.168.100.21
         Subnet mask: 255.255.255.0
         Gateway: 192.168.100.60
         Remote login: Enabled
...
   Controller in Slot B
      Status: Online
      Current configuration
         Firmware version: 06.12.03.00
```

```
            Appware version: 06.12.03.00
            Bootware version: 06.10.04.00
         NVSRAM version: N1742F900R912V06
      Pending configuration
         Firmware version: None
            Appware version: None
            Bootware version: None
         NVSRAM version: None
         Transferred on: None
      Board ID: 5884
      Product ID: 1742-900
      Product revision: 0520
      Serial number: 1T34058422
      Date of manufacture: 10. Oktober 2003
      Cache/processor size (MB): 1024/128
      Date/Time: Thu Jun 09 11:34:02 EDT 2005
      Associated Logical Drives (* = Preferred Owner):
         GPFS_data2*, GPFS_data4*, GPFS_metadata2*, GPFS_metadata4*
      Ethernet port: 1
         MAC address: 00:a0:b8:12:10:6e
         Host name: fast9itso2
         Network configuration: Static
         IP address: 192.168.100.22
         Subnet mask: 255.255.255.0
         Gateway: 192.168.100.60
         Remote login: Enabled
...
ARRAYS------------------------------
   Number of arrays: 8

   Array 1 (RAID 5)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Current owner: Controller in slot A
      Associated logical drives and free capacities:
         GPFS_data3 (271,463 GB)
      Associated drives (in piece order):
         Drive at Enclosure 1, Slot 1
         Drive at Enclosure 1, Slot 2
         Drive at Enclosure 1, Slot 3
         Drive at Enclosure 1, Slot 4
         Drive at Enclosure 1, Slot 5
   Array 2 (RAID 5)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Current owner: Controller in slot B
      Associated logical drives and free capacities:
```

```
         GPFS_data4 (271,463 GB)
      Associated drives (in piece order):
         Drive at Enclosure 1, Slot 6
         Drive at Enclosure 1, Slot 7
         Drive at Enclosure 1, Slot 8
         Drive at Enclosure 1, Slot 9
         Drive at Enclosure 1, Slot 10
   Array 3 (RAID 5)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Current owner: Controller in slot A
      Associated logical drives and free capacities:
         GPFS_data1 (271,463 GB)
      Associated drives (in piece order):
         Drive at Enclosure 2, Slot 1
         Drive at Enclosure 2, Slot 2
         Drive at Enclosure 2, Slot 3
         Drive at Enclosure 2, Slot 4
         Drive at Enclosure 2, Slot 5
   Array 4 (RAID 5)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Current owner: Controller in slot B
      Associated logical drives and free capacities:
         GPFS_data2 (271,463 GB)
      Associated drives (in piece order):
         Drive at Enclosure 2, Slot 6
         Drive at Enclosure 2, Slot 7
         Drive at Enclosure 2, Slot 8
         Drive at Enclosure 2, Slot 9
         Drive at Enclosure 2, Slot 10
   Array 5 (RAID 1)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Current owner: Controller in slot A
      Associated logical drives and free capacities:
         GPFS_metadata1 (67,866 GB)
      Associated drives (in piece order):
         Drive at Enclosure 2, Slot 11 [mirrored pair with drive at enclosure
2, slot 12]
         Drive at Enclosure 2, Slot 12 [mirrored pair with drive at enclosure
2, slot 11]
   Array 6 (RAID 1)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
```

```
      Current owner: Controller in slot B
      Associated logical drives and free capacities:
         GPFS_metadata2 (67,866 GB)
      Associated drives (in piece order):
         Drive at Enclosure 2, Slot 13 [mirrored pair with drive at enclosure
2, slot 14]
         Drive at Enclosure 2, Slot 14 [mirrored pair with drive at enclosure
2, slot 13]
   Array 7 (RAID 1)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Current owner: Controller in slot A
      Associated logical drives and free capacities:
         GPFS_metadata3 (67,866 GB)
      Associated drives (in piece order):
         Drive at Enclosure 1, Slot 11 [mirrored pair with drive at enclosure
1, slot 12]
         Drive at Enclosure 1, Slot 12 [mirrored pair with drive at enclosure
1, slot 11]
   Array 8 (RAID 1)
      Status: Online
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Current owner: Controller in slot B
      Associated logical drives and free capacities:
         GPFS_metadata4 (67,866 GB)
      Associated drives (in piece order):
         Drive at Enclosure 1, Slot 13 [mirrored pair with drive at enclosure
1, slot 14]
         Drive at Enclosure 1, Slot 14 [mirrored pair with drive at enclosure
1, slot 13]

STANDARD LOGICAL DRIVES------------------------------

SUMMARY
   Number of standard logical drives: 8
   See other Logical Drives sub-tabs for premium feature information.

   NAME            STATUS   CAPACITY    RAID LEVEL  ARRAY
   GPFS_data1      Optimal  271,463 GB  5           3
   GPFS_data2      Optimal  271,463 GB  5           4
   GPFS_data3      Optimal  271,463 GB  5           1
   GPFS_data4      Optimal  271,463 GB  5           2
   GPFS_metadata1  Optimal  67,866 GB   1           5
   GPFS_metadata2  Optimal  67,866 GB   1           6
   GPFS_metadata3  Optimal  67,866 GB   1           7
   GPFS_metadata4  Optimal  67,866 GB   1           8
```

```
DETAILS
   Logical Drive name: GPFS_data1
      Logical Drive ID: 60:0a:0b:80:00:12:0f:33:00:00:00:0f:42:87:55:00
      Subsystem ID (SSID): 52
      Status: Optimal
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Preferred owner: Controller in slot A
      Current owner: Controller in slot A
      Capacity: 271,463 GB
      RAID level: 5
      Segment size: 256 KB
      Modification priority: Lowest
      Associated array: 3
      Read cache: Disabled
      Write cache: Disabled
      Write cache without batteries: Disabled
      Write cache with mirroring: Disabled
      Flush write cache after (in seconds): 10.00
      Cache read ahead multiplier: 0
      Enable background media scan: Enabled
      Media scan with redundancy check: Disabled
...
   Logical Drive name: GPFS_metadata1
      Logical Drive ID: 60:0a:0b:80:00:12:0f:33:00:00:00:12:42:87:55:16
      Subsystem ID (SSID): 54
      Status: Optimal
      Drive type: Fibre Channel
      Enclosure loss protection: No
      Preferred owner: Controller in slot A
      Current owner: Controller in slot A
      Capacity: 67,866 GB
      RAID level: 1
      Segment size: 64 KB
      Modification priority: Lowest
      Associated array: 5
      Read cache: Enabled
      Write cache: Enabled
      Write cache without batteries: Disabled
      Write cache with mirroring: Enabled
      Flush write cache after (in seconds): 10.00
      Cache read ahead multiplier: 5
      Enable background media scan: Enabled
      Media scan with redundancy check: Disabled
...
```

B

Typical broadcast system workflow

This appendix provides a sample broadcasting environment.

Ingest System

The essence[1] (contents) for the media storage system is provided by different external sources, in different formats, such as real time (live) feeds of *SDI* video, video tape, video files from cameras equipped with optical disks or solid state memory.

Some systems also provide low-resolution browse versions of the video and descriptive metadata automatically.

Two distinct steps take place during the ingest process:

1. The essence should be stored, as fast as possible on the media storage system, preferably up to ten times faster than real time. Possibly, the essence has to be encoded, or trans-coded, to the desired video and/or audio format. A *UMID* has to be provided by the source or generated at the ingest phase. This step could require a separate specific ingest storage system as the requirements for this functionality could differ substantially from the characteristics of the general media storage.

[1] The term “essence” is used in broadcasting environments to name the actual video/audio data (content).

2. The metadata describing the essence has to be collected, generated and introduced into the metadata repository of the production metadata management system. This could be done manually or automatically. This also includes the generation of browse resolution version, thumbnails generated by key frame extraction and textual annotation of the essence.

This second functionality also provides essentially the archiving function, although perhaps executed at a different stage in the production process. However, the same process and tools can be used at a later stage to perform archiving or more detailed annotation of any material already available on the general media storage.

Integration of editing tools

Metadata for the essence is available in the form of textual description, but also as a low-resolution video version of the material itself. Therefore, video can be viewed, browsed and even an elementary editing can be performed by creating an EDL (editing decision list) based on frame accurate low-resolution video. This EDL could be processes by a rendering engine to produce the high-resolution essence.

Also, the essence can be manipulated directly by craft editing tools and stored back on the central storage.

Ideally, the browse and craft editing systems should access the media material only via the production metadata management system. That means that as described above, the generic metadata browser should have the ability to launch the proper editing tool with the selected media file as parameter, and the 'open' and 'save' functions within these editing tools should invoke again the production metadata management system to pass check-in and check-out of files back to this system. This should keep the metadata repository consistent.

Possibly, the ingest system should be invoked if new material is created, to generate the appropriate metadata, browse video and annotation.

On-Air Storage

Broadcasting from video-file storage still has different requirements in terms of timing, availability, etc., than the general media storage system. Most likely, specific on-air storage equipment will have to be selected for this functionality.

Essence has to be transported to this on-air system from the general media storage platform. This should happen as fast as possible via file transport over IP. If no separate platform is necessary, virtual transport could be imagined via virtual links, or snapshot techniques.

Hierarchical Storage Management (On-Line/Near-Line/Off-Line)

Complementary to the online hard disk storage architecture for immediate access to media essence files, in the total media storage model there is a place and a need for a secondary storage system based on a fundamentally different technology.

Whatever mix between on line disk-based and near line data-tape based storage the eventual media storage model will be, there will always be a requirement for off line storage for backup and disaster recovery reasons. By off line, remote storage of the media carrier is meant, e.g., data-tape stored remotely in a location outside the origin building.

Data-tape could be and probably is the most evident choice for backup technology, although perhaps optical disk based technology should be looked into also. Advantage of using the same data-tape technology is that no new investments are required in terms of play out hardware, as the near line data-tape robot infrastructure is already available. Disadvantage could be that exactly the same technology is used as for the near line storage carrier, so one doesn't have the benefit of an independent type of media carrier.

From the point of view of the client which accesses the media essence through the generic file system the place where the media file is stored should be irrelevant, whether it is stored on the hard-disk based on line storage system, the near line datatape based storage system, or even the off line backup system. Only the physical access time will be different. So, the file system should shield or hide these hierarchical storage levels from the client. In fact, what is described here is exactly the behavior of a typical hierarchical storage management system, where the physical location of every media essence file should be managed in the background, transparently to the view that the client has on the file system.

Ingest Function

Essence should be ingested into the central media storage system, so that the media is in principal available to every connected station, evidently within the constraints of the applied security context. As a target, the ingest system should be capable of ingesting at least ten simultaneous feeds, independent of their format. This could be real time live feeds, video-tape or video files. In the case of non real time video feeds, ingest should be as fast as possible, preferably in the order of ten times faster than real time.

This will have a profound influence on the question whether the central general storage system is capable of ingesting this throughput and to what extent, or whether there is need for a specific storage system with specific technical characteristics which allow this system to accept this throughput and to what

extent such a system can scale the number of simultaneous inputs and at which speed.

During ingest, trans-coding to the proper format will be necessary if the video input is not already delivered as a file in the desired format. Trans-coding to low-resolution material in parallel to ingesting the essence is preferable, as is the generation of thumbnails by key frame extraction.

EDL Generation/Rendering

Many video-editing operations can be performed in the low-resolution area, using frame accurate video browsers. The resulting EDLs should be passed to rendering engines to execute the EDL commands and process the essence in the high-resolution area. The result should again be 'ingested' in the media storage system and in the production material metadata management system.

Broadcasting from 'On-Air' Storage System

Just as the ingest storage could be a separate storage system, because of the specific ingest throughput requirements, so could the on-air storage from which can be broadcasted put some stringent requirements on the architecture that result in a specific broadcast server platform. Transfer of essence to this platform should again be performed as fast as possible, preferably over a high bandwidth IP network. Integration of the broadcast storage in the central generic media storage system should on the other hand make this transfer of essence unnecessary. The POC should try to verify if this last model is technically feasible, and if not what the specific requirements are that have to be fulfilled by a specific on-air storage system.

Media Access via 'Generic Browser' (Hiding Technical File System)

The file system view is the most technical access method to the media essence. Since the media file system will contain all media files, a user-friendly directory structure and categorization of files suitable for every function is not obvious. It is certainly not the proper tool to generate user- or roll-based views, let alone user-customized views. Probably, the file system security alone is not elaborate enough to provide out of the box the desired user- and media-file protection between different departments, programs, etc.

Also, since the media files should be uniquely labelled, the UMID could be a useful candidate as file-name, making the filename however unreadable for the human user.

Therefore, the user access to the media files should not be done via the file system view directly, but via a generic browser based on the descriptive metadata of the media files. This would hide the technical file system and any technical naming conventions from the user, and allow for a user- and or

roll-based customizable view. This would also provide a higher-level security context on the media access.

Metadata Managed by 'Production Material Management System'

The descriptive metadata, not essential for the technical media production of the essence should reside and be managed in a 'production metadata management system'. Since this metadata can be treated as formatted documents, preferably XML structured documents, a document management system could probably provide this functionality.

On the other hand, the metadata that is essential for the production of the essence (structural metadata) could be stored in the MXF or AAF file-container of the essence on the central media storage system.

Media Files are not stored in Metadata Repository

Typical document management systems store the documents themselves also in their repository. Since the essence files can be extremely large, this would place a to heavy burden on the document management system. Therefore, only a reference or pointer should be stored in the metadata repository, while the media file itself is physically stored on the central media storage. The actual physical storage location of the file, whether it is stored on the disk-based online storage, on the near-line storage, or even stored off-line should be hidden from and transparent to the metadata management system. It is the task of the hierarchical storage management system to resolve the pointer to the exact physical location and to retrieve the media file from the correct storage level or place.

Integration of 'Open' - 'Save' Function of Media Application'

Since the generic metadata browser should be the single access point of the media files, the open and save functions of the relevant media applications, like browse and craft editors, should interface with the production metadata management system, and not providing direct access to the underlying file system structure.

Similar or More Advanced Search and Retrieve Functionality than BASIS+

The functionality of the generic metadata browser and of the production metadata management system should be enhanced by a intelligent search and retrieve, at least equivalent to the present BASIS+ functionality. Further enhancements can be envisaged in a further stage of the POC. A similar requirement will be included in the engineering POC, so probably the same development could cover both areas.

Archiving/Annotation Functionality

Metadata should be ingested at the same time during the ingest of the essence, linking the media material on the media storage to the production material metadata management system. This metadata ingest should be provided in an automated way if the metadata is already created or available in the camera system, or should be introduced manually.

Annotation could also be performed at the moment of ingesting if time is available and if this is technically possible, e.g., taking into account the faster than real time ingest rate. This function is similar to the archiving function, although the moment of annotation is perhaps different with relation to the present archiving workflow or process. Annotation at a later stage could make use of the same functionality provided by the ingest system.

Cache director

The cache director could also be considered as part of the storage front-end,

since its tries to service the request coming from the hosts out of the cache buffers. It typically divides its total memory between read and write cache. It serves as an intermediate layer between the host and the physical storage. Since physical access to the disk system is much slower than an I/O to memory it tries to service as much host requests as possible from and to its cache memory, and evades as much as possible any physical access to the disk system.

If read cache is enabled, which is the default on most systems, when it receives a read request from a host, it first looks into the read cache whether the requested data resides in memory. If so, the storage controller can send the requested data straight to the host without performing any read request to the physical disk system. This is called a ‘cache-hit’. If the request is not found in the read cache memory, a ‘cache-miss’ occurs. The read request is passed to the disk director and a physical read from disk is performed. The requested data is stored in the read cache and is also sent to the host. If the same data is requested again shortly afterwards, the chances are high that the data is still present in the read cache, so that the request can be services this time out of the cache. A cache-hit results of course in a much faster response than a cache-miss, where a physical read operation from the disk takes place.

Since chances are high that the next request from the host will be the next data block on the disk system, the cache director tries to optimize its physical access to the disk system in order to reach a maximum cache-hit/miss ratio. Therefore with each physical read to the disk system, the cache director requests also the next few blocks from the disk system to be read. It executes a so-called ‘prefetch’. It stores the extra data in the cache, hoping that the host will indeed request this data next, so that it then doesn’t have to perform again a physical disk read. If the

read cache is full, it will drop the oldest or least used data using some sort of intelligent algorithm, and overwrite this with newly read data.

However, if the access pattern is such that cache-hits are very rare, the prefetch algorithm fails its purpose and will on the contrary generate surplus load on the disk system. This could lead to a decrease in throughput.

If write cache is enable, the cache director tries to store the data from a write request form the host into the write cache buffer. Is this succeeds, that is if there is sufficient space to store the data, the controller sends a status response to the host that the data is being stored, before it physically is written to disk. This response is of course much faster than when the host has to wait for the controller to signal that the data resides physically on the disk. The cache director tries to de-stage the physical write operation in time to a moment when the system is less busy, or to spread it in time if there is enough free throughput capacity to write to the disk. So, again using an intelligent algorithm, it purges slowly the write cache to the disk system to free up space. If however the hosts write so much data to the storage system that the write cache is getting filled up, the cache director first has to empty part of the write cache to the disk system, before it can accept the data form the host and store it in the write cache buffer. In this case, the write operation can even take longer than when the data was directly written to disk in the first place. The cache director uses all sorts of threshold limits to start and stop flushing the cache.

There is a possible enhancement that the read cache director can also look for cache-hits in the write cache memory, thereby increasing the cache hit/miss ratio. In that case, the cache director has an extra incentive to keep the write data as long as possible in its cache buffers, even if the data has been written to the disk system. However, most storage system cannot service read request out of their write cache, and even if they can, they have to copy the data from the write cache to the read cache buffer first, and then send the data to the host.

Since the cache memory is not permanent, keeping data in the cache buffer brings certain risks with it. If the cache memory or the controller would fail, the system would loose the data in its cache. This is not critical for the read cache, since that data resides safely on the permanent disk system. However, the data in the write cache is not yet been stored on disk, so it is permanently lost. Several counter measures can be made. First, the cache memory buffer is kept under tension by batteries. This has an limited life span, so if the controller is not brought back on line fast enough before the battery is depleted the data is still lost. Another way to circumvent this from happening is mirroring the write cache in the second controller. That is, every write operation to the cache of the first controller is also transferred to the second controller and written in its write cache. Only then, the first controller sends the response to the host that the data is written. This delays the write operation slightly, but it is still faster than writing to

the physical disk. In case of cache or controller failure, the second controller still has all the data intact and can perform the final physical write. The disadvantage is that only half of the write cache memory can be actively used since the second half mirrors the write cache of the other controller, and vice versa. The way the cache is mirrored differs also from system to system. Many storage systems have dedicated FC links between the controllers specially used for synchronization of the controllers and mirroring of the write cache. Others, like the FAStT900 use the back-end loops for the mirroring and contend therefore for the bandwidth together with the disks. This has negative impact on the total throughput capacity of the storage system.

Cache can also been disabled, in which case every write is directly passed to the physical disk system, thereby delaying the response of the controller to the host to the moment when the data is effectively residing safely on the disk. Depending on the type of load, this could result in a more constant response behavior of the system. This is most probably the case in the media operational environment, since there very large file are written at a sustained rate, which probably will continuously overflow the write cache buffer of the controllers.

C

Yam - installation server setup tool

Yam is a tool that creates a repository and an installation server, based on Red Hat or SUSE ISO images and downloads updates from the Internet. It can be used together with tools such as apt, yum or yast for downloading and installing new packages or upgrading your system.

Yam comes with a default yam configuration file, /etc/yam.conf, and has more example config files as part of the yam package.

Example: C-1 Example Yam config file for Red Hat Enterprise Linux 4

```
[main]
srcdir = /var/yam
wwwdir = /var/www/yam
metadata = apt createrepo

[rhel4as]
name = Red Hat Advanced Server $release U1 ($arch)
release = 4
arch = i386 x86_64 ppc
iso = RHEL$release-U1-$arch-AS-disc?.iso
updates = file:///var/spool/up2date/
dag = rsync://apt.sw.be/pub/freshrpms/pub/dag/redhat/el4/en/$arch/RPMS.dag/
```

This configuration file will expect the RHEL4 ISO files to be located in /var/yam (or /var/yam/rhel4as-i386 if you want) and will create a complete repository and installation server in /var/www/yam. (In this case for both i386™, x86_64 and ppc architectures if you have those ISOs available).

Furthermore it will allow you to create a repository of the locally archived updates and a remote repository of additional RHEL software.

Now you can start synchronizing the remote repositories locally by doing:

```
yam -uxvv
```

And you can generate the repository by doing:

```
yam -gvv
```

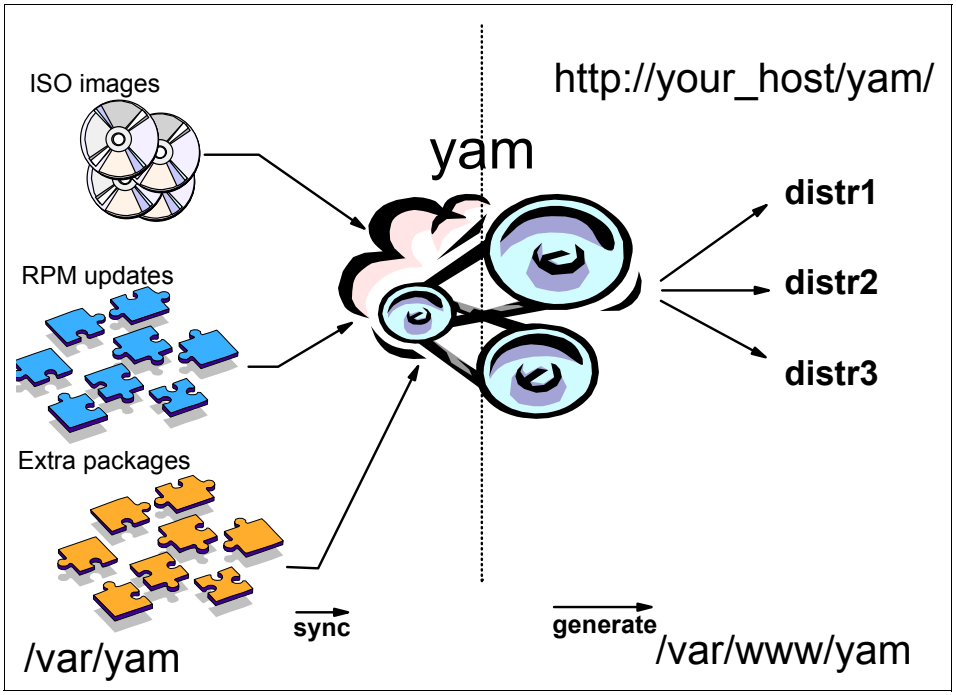

Figure C-1 Yam server diagram

Yam will provide more information the more -v options you provide. If all went well, you can browse through the installation server (and ISO images) by doing:

```
find /var/www/yam/ -type d | less
```

For SLES9 there are a few extra steps required that Yam will not take care of (yet). These extra steps are explained in the documentation that comes with the

Yam package. Even though Yast allows you to set up an installation server too, it does this by copying the complete content of the ISO files, Yam will not copy the content (reducing the required diskspace) but will mount the ISOs locally and create symlinks. Doing this, you can find all available packages merged in /var/www/yam/rhel4as-i386/RPMS/ or look at the other repositories for a different view of what is available.

More information about Yam is available from the Yam Web site at:

http://dag.wieers.com/home-made/yam/

Abbreviations and acronyms

eRMM	Enterprise Removable Media Manager
ACL	Access Control List
ADT	Auto Logical-Drive Transfer
AVT	Auto Volume Transfer
BIOS	Basic Input Output System
CSM	Cluster System Management
DM	Digital Media
FC	Fibre Channel
GPFS	General Parallel File System
HACMP	High Availability Cluster Multi-Processing
HBA	Host Bus Adapter
HDTV	High Definition Television
HSM	Hierarchical Storage Management
IBM	International Business Machines Corporation
ITSO	International Technical Support Organization
LAN	Local Area Network
LUN	Logical Unit Number
MAN	Metropolitan Area Network
MAN	Metropolitan Area Network
MPP	Multi Path Proxy
NFS	Network File System
NLE	Non-Linear Editing
NSD	Network Shared Disk
NVSRAM	Non Volatile Storage Random Access Memory
OESV	Open Enterprise System Virtualization
OS	Operating System
PPRC	Peer to Peer Remote Copy
RAID	Redundant Array of Independent Disks
RDAC	Redundant Dual Active Controller
ROI	Return of Investment
RPC	Remote Procedure Call
SAN	Storage Area Network
SDTV	Standard Definition Television
SMB	Shared Message Block
SOL	Serial over LAN
TCO	Total Cost of Ownership
VGDA	Volume Group Descriptor Area
VoD	Video on Demand
VSD	Virtual Shared Disk
WAN	Wide Area Network

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 246. Note that some of the documents referenced here may be available in softcopy only.

- *Tuning IBM @server xSeries Servers for Performance*, SG24-5287

Other publications

These publications are also relevant as further information sources:

- *General Parallel File System (GPFS) for Clusters: Concepts, Planning, and Installation*, GA22-7968
- *General Parallel File System (GPFS) for Clusters: Administration and Programming Reference*, SA22-7967
- *AIX 5L Version 5.3 Performance Management*, SC23-4905

Online resources

These Web sites and URLs are also relevant as further information sources:

- IBM Digital Media

  http://www.ibm.com/solutions/digitalmedia

- IBM Enterprise Removable Media Manager

  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10358
  http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/FLASH10365

- GPFS research home page

  http://www.almaden.ibm.com/cs/shark/

- GPFS home page

  http://www.ibm.com/servers/eserver/clusters/software/gpfs.html

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

Redbooks

Configuration and Tuning GPFS for Digital Media Environments

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Configuration and Tuning GPFS for Digital Media Environments

An introduction to Digital Media environment

GPFS configuration internals revealed

Installing and tuning your environment

In today's environments, when access to information is more and more demanding, media companies are implementing new solutions for their environments, focusing on performance, availability, and cost.

Because GPFS historically comes from multimedia roots, the adoption of this product for digital media environments comes as a natural evolution.

This IBM Redbook focuses on the newest GPFS functionality and its on demand characteristics related to (but not limited to) digital media environments.

Digital media information flow and requirements are analyzed and matched with GPFS functionality. Interoperability of a GPFS cluster and file sharing applications for Windows clients are discussed.

This redbook presents a nine step approach for configuring a GPFS cluster, and also presents various tools and techniques that make it easier for a systems administrator in GPFS implementation and day-to-day management tasks.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6700-00 ISBN 0738492787