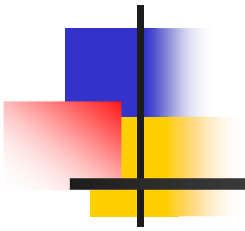
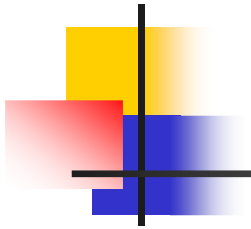


An Introduction to GPFS

Vladimir Sapunenko
INFN-CNAF

Vladimir.sapunenko@cnaf.infn.it



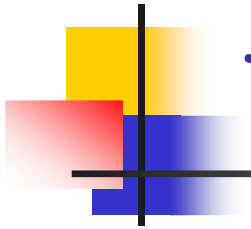
Outline

Part 1: General Introduction

- Necessity of Parallel I/O
- Mostly known implementations
- GPFS features and advantages
- Architecture review

Part 2: GPFS Installation, Configuration and Administration

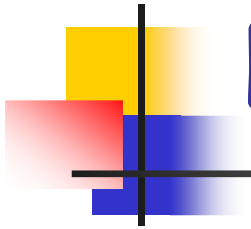
- GPFS features in depth
- Installation
- Administration
- Configuration examples



There are a lot of ways to do disk I/O.

Since we are interested in HPC I/O in general and parallel I/O in particular, lets first define parallel disk I/O and then briefly survey different paradigms of disk I/O used on HPC environment.

Assume that term "I/O" in this presentation refers to disk I/O (unless stated otherwise).

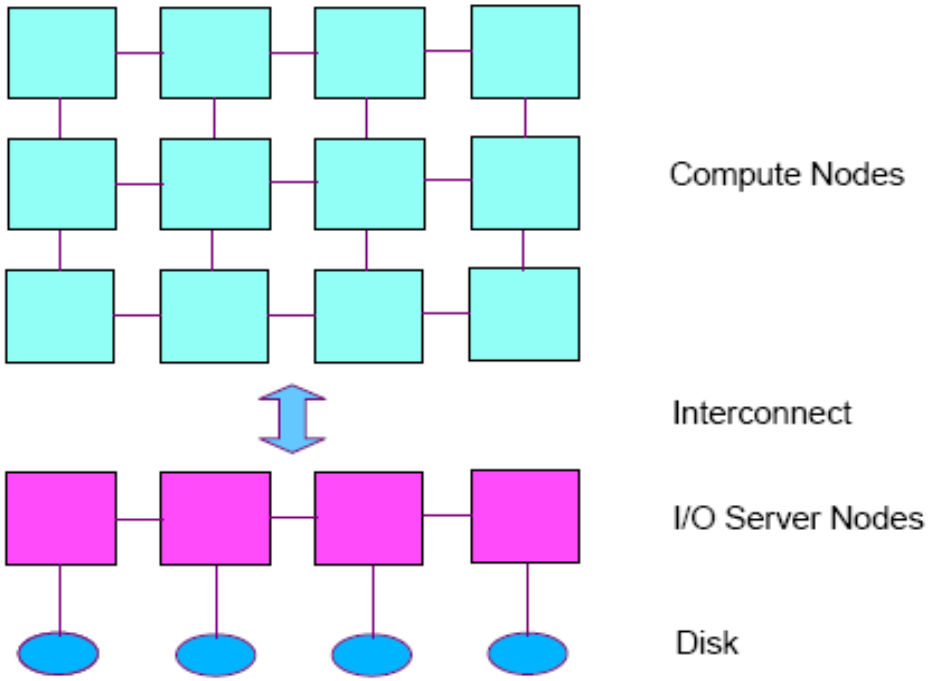


Definition of Parallel I/O

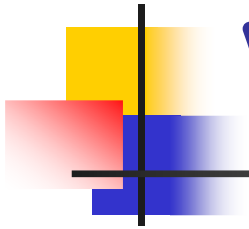
Referred to Ewing ("Rusty") Lusk of ANL:

- Multiple processes (possibly on multiple nodes) participate in the I/O
- Application level parallelism
- "File" is stored on multiple disks on a parallel file system
- Additional interfaces for I/O (MPI)

What is Parallel I/O?



COMMENT:
As HPC systems have matured, the models for storage I/O have become more varied and complex.



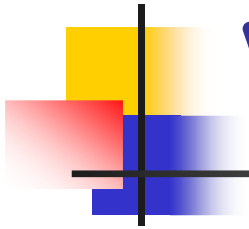
What is Parallel I/O?

Parallel I/O should safely support:

- application level I/O parallelism across multiple computational nodes
- physical parallelism over multiple disks and servers
- parallelism in file system overhead operations

A parallel file system must support:

- Parallel I/O
- consistent global name space across all nodes of the cluster
  - this includes maintaining a consistent view across all nodes for the same file
- programming model allowing programs to access file data
  - distributed over multiple nodes
  - from multiple tasks running on multiple nodes
- physical distribution of data across disks and network entities
- eliminates bottlenecks both at disk interface and network, providing more effective bandwidth to I/O resources

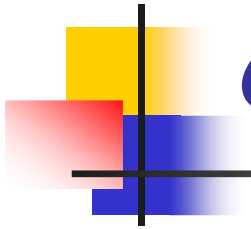


What is Parallel I/O?

The promise of parallel I/O is increased performance and robustness. And it is more natural in multi-node HPC system.

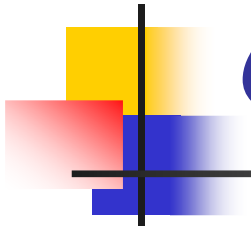
The challenge of parallel I/O is that it is a more complex model of I/O to use and manage.

Storage Architecture classification



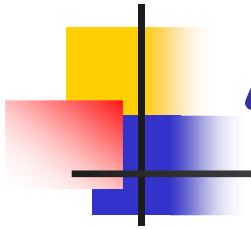
The following pages examine an architectural classification of storage I/O architectures commonly used in HPC systems. They support various degrees of parallel I/O and do not represent mutually exclusive choices.

- **Conventional I/O**
- **Asynchronous I/O**
- **Network File Systems**
- **Basic Parallel I/O**
  - Single Component Architecture
- **Centralized Metadata Server with SAN Attached Disk**
  - Dual Component Architecture
- **High Level Parallel I/O**
  - Triple Component Architecture
- **Recent Developments**



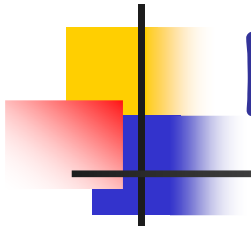
Conventional I/O

- Local file systems
- Basic, "no frills, out of the box" file system
- Journal, extent based semantics
  - journaling: to log information about operations performed on the file system meta-data as atomic transactions. In the event of a system failure, a file system is restored to a consistent state by replaying the log and applying log records for the appropriate transactions.
  - extent: a sequence of contiguous blocks allocated to a file as a unit and is described by a triple consisting of <logical offset, length, physical>
- If they are a native FS, they are integrated into the OS (e.g., caching done via VMM)
- Intra-node process parallelism
- Disk level parallelism possible via striping
- Not truly a parallel file system
- Examples: Ext3, JFS, XFS



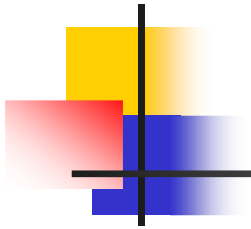
Asynchronous I/O

- Abstractions allowing multiple threads/tasks to safely and simultaneously access a common file
- Built on top of a base file system
- Parallelism available if its supported in the base file system
- Part of POSIX 4, but not supported on all unix based file systems (e.g., Linux 2.4, but Linux 2.6 now includes it)
- AIX, Irix, Solaris support AIO



Networked File System (NFS)

- Disk access from remote nodes via network access (e.g., TCP/IP over Ethernet)
- NFS is very widespread and the most common example
- it is not truly parallel
  - old versions are not cache coherent (is V3 truly safe?)
  - write requires O_SYNC and -noac options to be safe
- poorer performance for I/O intensive HPC jobs
  - write: only 90 MB/s on system capable of 400 MB/s (4 tasks)
  - read: only 381 MB/s on system capable of 740 MB/s (16 tasks)
- uses POSIX I/O API, but not its semantics
- useful for on-line interactive access to smaller files
- while NFS is not designed for general parallel file access on an HPC system, by placing restrictions on an application's storage I/O model, some users get "good enough" performance from it



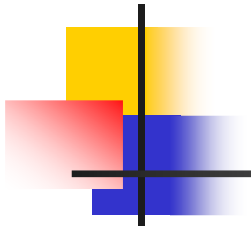
Basic Parallel I/O

Single Component Architecture

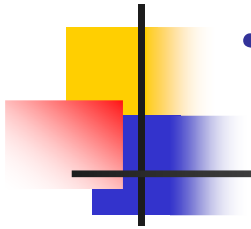
- Satisfies basic definition of parallel I/O
- Parallelizes file, metadata and control operations
- POSIX I/O model with extensions
  - byte stream using API with read(), write(), open(), close(), lseek(), stat(), etc.
  - extends POSIX model to support safe parallel data access semantics
    - these options guarantee portability to other POSIX based file systems for applicationsusing the POSIX I/O API
  - generally has API extensions, but these compromise portability
- Good performance for large volume, I/O intensive jobs
- Works best for large block, sequential access patterns, but vendors can add optimizations for other patterns
- Example: GPFS, GFS (Sistina/Redhat)
  - GPFS has very flexible cluster/disk connectivity allowing large node count scaling

Centralized Metadata Servers with SAN Attached Disk

Dual Component Architecture



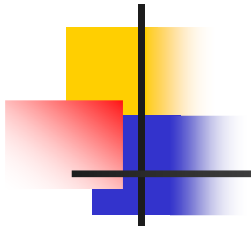
- Parallel user data semantics, but non-parallel metadata semantics
  - Support POSIX API, but with parallel data access semantics
- Metadata maintained and accessed from a single common server
  - Failover features allow a backup metadata server to takeover if the primary fails
  - Uses Ethernet (100 MbE or 1 GbE) for metadata access
  - Potential scaling bottleneck??
- All "disks" connected to all client nodes via the SAN
  - file data accessed via the SAN, not the node network
    - removes need for expensive node network (e.g., Myrinet)
  - inhibits scaling due to cost of FC Switch Tree (i.e., SAN)
- Ideal for smaller numbers of nodes
  - SNFS advertises up to 50 clients (is this reality?)
  - CXFS scales only to 10-12 servers from some users
- Example: CXFS (SGI), SNFS (ADIC), SanFS (IBM)
  - These may be called "parallel like" file systems since metadata processing is not parallel



Triple Component Architecture

Lustre and Panasas, are 2 recently developed HPC style parallel file systems which began "from a clean sheet of paper" in their design that distinguishes them from other file systems in this classification. They have a number of architectural similarities.

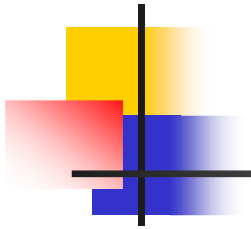
- **3 component architecture**
  - storage clients
  - storage servers
  - metadata servers (e.g., metadata)
- file data access over the node network between storage clients and servers (e.g., GbE, Myrinet)
- **Object oriented architecture**
  - object oriented disks are not generally available yet, so the current implementation is in SW and not fully generalized
  - OO design is blind to the application (i.e., uses POSIX API with parallel semantics)
- **Designed to facilitate storage management (e.g., virtualization for SanFS)**
- **Focus on Linux/COTS environments**



Higher Level Parallel I/O

- High level abstraction layer providing parallel model
- Built on top of a base file system (conventional or parallel)
- MPI-I/O is the most common model
  - parallel disk I/O extension to MPI in the MPI-2 standard
  - semantically richer API
  - portable
- Requires significant source code modification for use in legacy codes, but it has the advantages of being a standard (e.g., syntactic portability)

Which Architecture is Best?



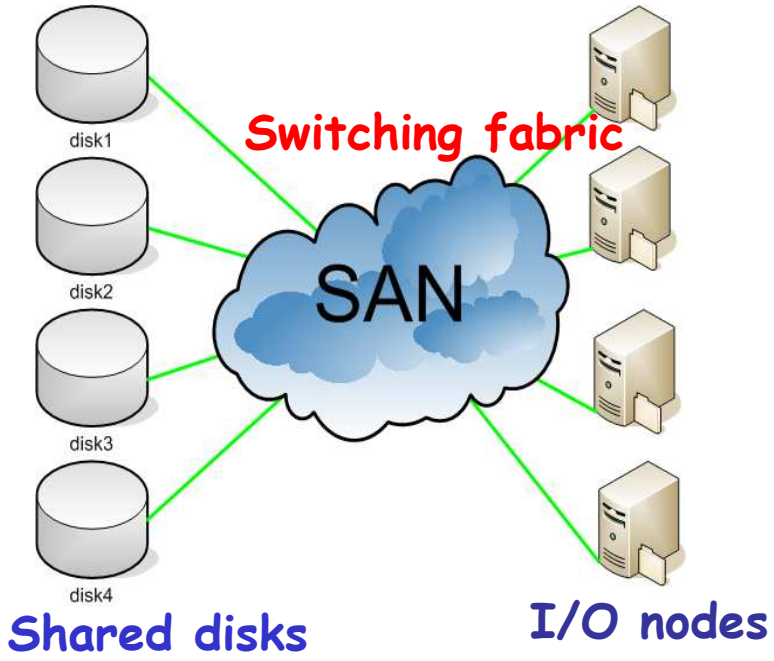
There is no immediate answer to this question.

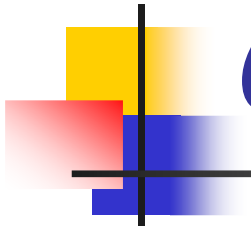
- It is application/user specific.
- All of them serve specific needs.
- All of them work well *if properly deployed and used according to their design specs*.
- Issues to consider are
  - application requirements
  - often requires compromise between competing needs
- how the product implements a specific architecture

What is GPFS?

GPFS= General Parallel File System
IBM's shared-disk, parallel cluster file system.

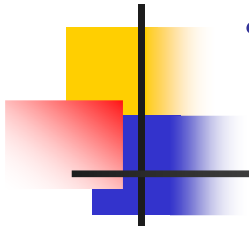
- *Cluster*: 2300+ nodes (tested), fast reliable communication, common admin domain
- *Shared disk*: all data and metadata on disk accessible from any node through disk I/O interface (i.e., "any to any" connectivity)
- *Parallel*: data and metadata flows from all of the nodes to all of the disks in parallel
- *RAS*: reliability, accessibility, serviceability
- *General*: supports wide range of HPC application needs over a wide range of configurations





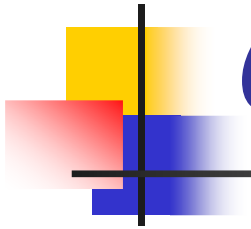
GPFS Features

- General Parallel File System
  - mature IBM product generally available for 10 years (GPFS has been available on AIX since 1998 and Linux since 2001)
- Works on AIX and Linux
- Adaptable to many user environmnets by supporting a wide range of basic configurations and disk technologies
- Provides safe, high BW access using the POSIX I/O API
- Provides non-POSIX advanced features
  - e.g., DMAPI, data-shipping, multiple access hints (also used by MPI-IO)
- Provides good performance for large volume, I/O intensive jobs
- Works best for large record, sequential access patterns, has optimizations for other patterns (e.g., strided, backward)
- Converting to GPFS does not require application code changes provided the code works in a POSIX compatible environment



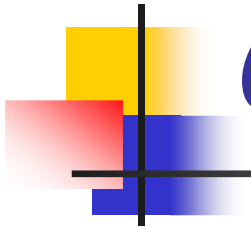
The file system

- Built from a collection of disks
- Each disk can contain data and/or metadata
- Supports quotas, snapshots and extended ACLs
- Coherency and consistency maintained via distributed lock manager



GPFS Features

- striping
- large blocks (with support for sub-blocks)
- byte range locking (rather than file or extent locking)
- access pattern optimizations
- file caching (i.e., pagepool) that optimizes streaming access
- prefetch, write-behind
- multi-threading
- distributed management functions (e.g., metadata, tokens)
- multi-pathing (i.e., multiple, independent paths to the same file data from anywhere in the cluster)

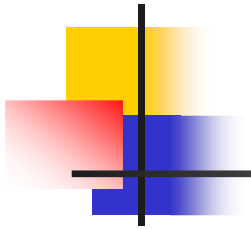


GPFS Features

GPFS provides many of its own RAS features and exploits RAS features provided by various subsystems

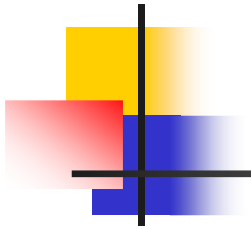
- If a node fails providing GPFS management functions, an alternative node assumes responsibility, reducing risk of loosing the file system.
- When using dedicated NSD servers with "twin tailed disks", specifying primary and secondary nodes lets the secondary node provide access to the disk if the primary node fails.
  - WARNING: internal SCSI and IDE drives are not twin tailed
- In SAN environment, failover reduces risk of lost access to data.
- GPFS on RAID architectures reduces risk of lost data.
- Online and dynamic system management allows file system modifications without bringing down the file system.
  - mmadddisk, mmdeldisk, mmaddnode, mmdelnode, mmchconfig, mmchfs

Information Lifecycle Management (ILM)



- policy-driven automation and tiered storage management
- storage pools
  - allow create groups of disks within a file system
  - new feature in v. 3.2 - external storage pools (to interact with an external storage management, e.g. TSM)
- filesets
  - sub-tree of the file system namespace
  - administrative boundary to set quotas
  - control initial data placement or data migration
- user-defined policies

Information Lifecycle Management (ILM)



- If the storage pool named "**system**" has an occupancy percentage above 90% now, bring the occupancy percentage of "**system**" down to 70% by migrating the largest files to storage pool "**data**":

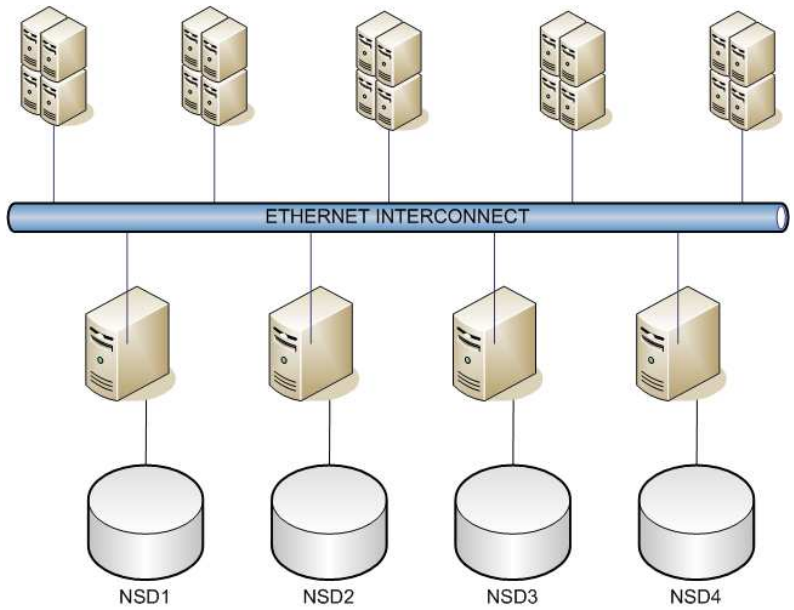
```
RULE 'mig1' MIGRATE FROM POOL ‘system'
THRESHOLD(90,70) WEIGHT(KB_ALLOCATED) TO POOL ‘data'
```

- Delete files from the storage pool named "**system**" that have not been accessed in the last 30 days, and are named like **temporary** files or appear in any directory that is named **tmp**:

```
RULE 'del1' DELETE FROM POOL ‘system' WHERE
(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME) > 30)
AND (lower(NAME) LIKE '%.tmp' OR PATH_NAME LIKE
'%/tmp/%')
```

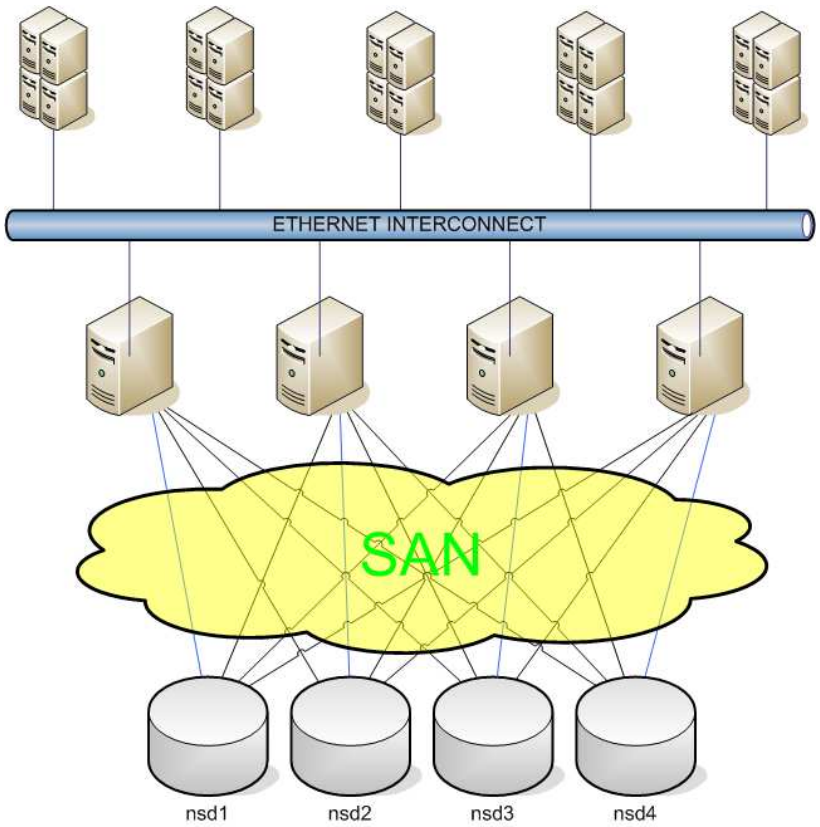
GPFS on DAS
(Directly Attached Storage)

- The most economic solution
- BUT there are some drawback:
  - To protect data from a single server failure need replicate data and metadata -> reducing usable space by a half
  - bus to disk ~320MB/s (2.5Gb/s)

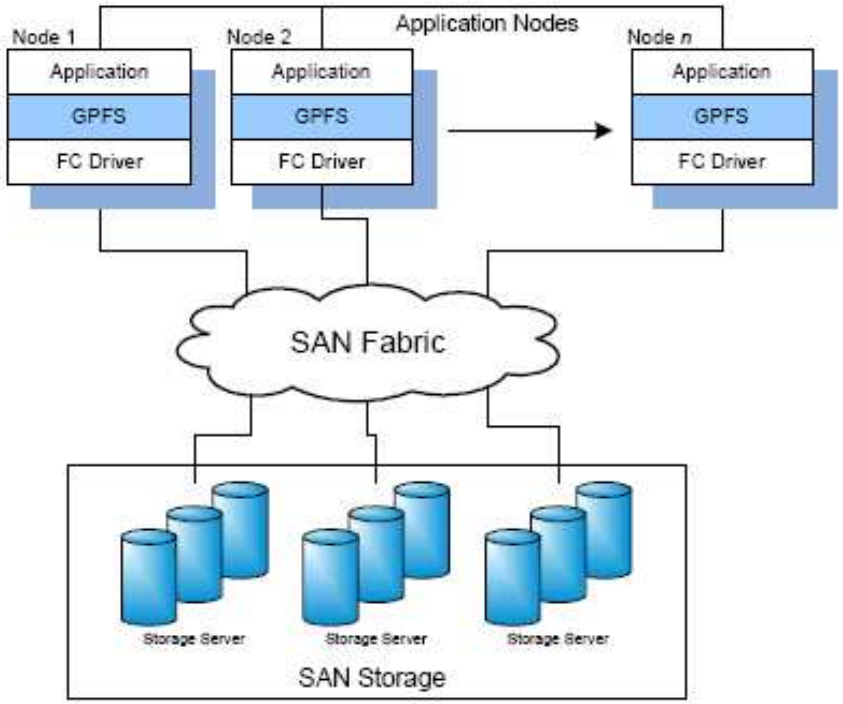


GPFS on SAN
(Storage Area Network)

- The most natural way to use GPFS
- All servers accessing all disks
- Failure of a single server will only reduce available bandwidth to storage by factor N-1/N (N – number of diskservers)
- Bandwidth to disks can be easily increased to 8Gb/s (with 2 dual cannel FC2 or with 1 dual channel FC4 HBA)

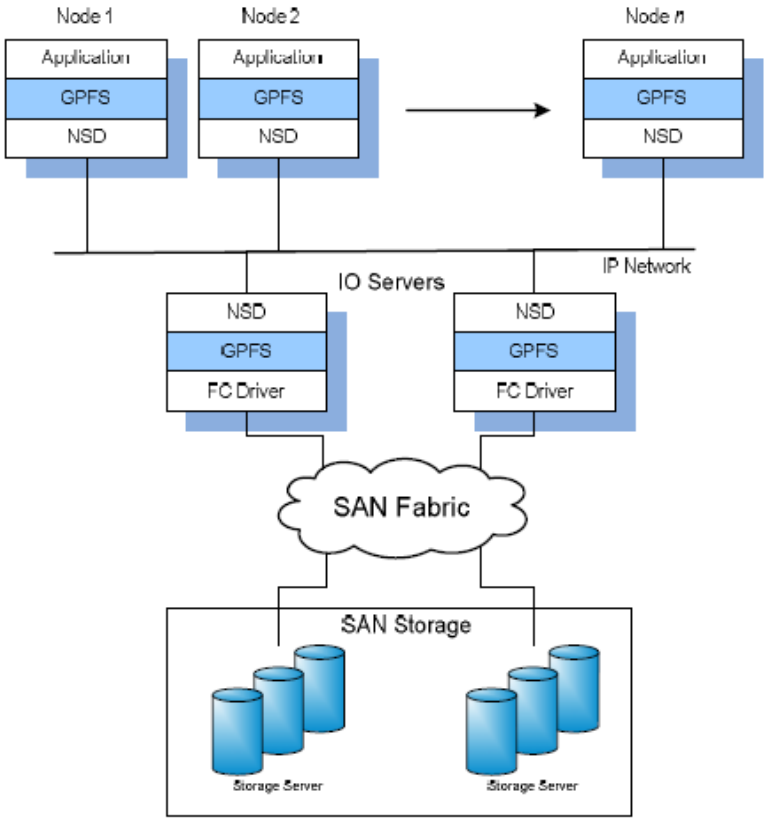


Cluster configuration

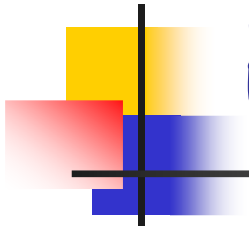


- Shared disk cluster (the most basic environment)
  - SAN storage attached to all nodes in the cluster via FiberChannel (FC)
  - All nodes interconnected via LAN
  - Data flows via FC
  - Control info transmitted via Ethernet

Network based block I/O



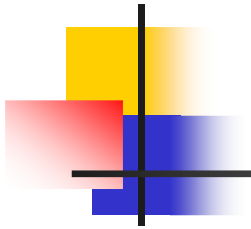
- Block level I/O interface over network – Network Shared Disk (NSD)
- GPFS transparently handles I/O whether NSD or direct attachment is used
- Intra-cluster communications can be separated using dedicated interfaces



Enhancements in GPFS v3.1

- The requirements for IP connectivity in a multi-cluster environment have been relaxed.
  - In earlier releases of GPFS, 'all-to-all' connectivity was required. Any node mounting a given file system had to be able to open a TCP/IP connection to any other node mounting the same file system, irrespective of which cluster either of the nodes belonged to.
- Enhanced file system administration
  - The "mmmount" and "mmumount" commands are provided for cluster-wide file system management.
  - Performans monitoring

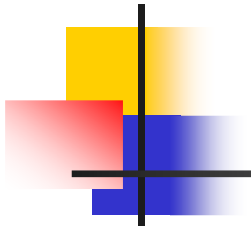
What's new in GPFS Version 3.2



- Performance enhancements
  - Defragmentation of a file system can now be run in parallel on all nodes in a cluster
  - Maximum pagepool size is increased to 256 GB
  - Fine grained directory locking improves multi-node file creation performance into a single directory
- native InfiniBand protocol support
  - GPFS uses RDMA to transfer data directly between the NSD client memory and the NSD server memory instead of sending and receiving the data over a TCP socket
- Monitoring
  - SNMP is supported for monitoring a GPFS cluster and generating notification traps
- rolling upgrades
  - install new GPFS code one node at a time without shutting down GPFS on other nodes
- up to eight NSD servers to simultaneously serve I/O requests to a single disk
  - different servers to serve I/O to different non-intersecting sets of clients

What's new in GPFS Version 3.2 (continued)

- Connection retries
  - If the LAN connection to a node fails GPFS will automatically try and reestablish the connection before making the node unavailable
- extended ILM functionality
  - single set of policies can be used to move data across different storage pools within a file system and move data between internal and external storage pools (disk ←→ tape)
  - pool thresholds
- Clustered NFS
  - high availability solution for NFS exporting file systems
  - Monitoring
  - Failover
  - Load balansing

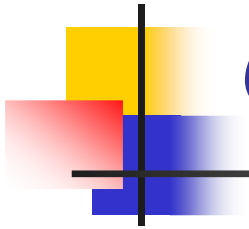


External Pools

- **External pools are really interfaces to external storage managers, e.g. HPSS or TSM**
  - External pool "rule" defines script to call to migrate/recall/etc. files

```
RULE EXTERNAL POOL 'PoolName' EXEC
  'InterfaceScript' [ OPTS 'options']
```

- **GPFS policy engine builds candidate lists and passes them to external pool scripts**
- **External storage manager actually moves the data**



Clustered NFS

GPFS Version 3.2 offers Clustered NFS which is a set of features and tools that support a high availability solution for NFS exporting file systems.

- some or all of the nodes in the GPFS cluster can export the same file systems to the NFS clients. (supported for systems with Linux only)

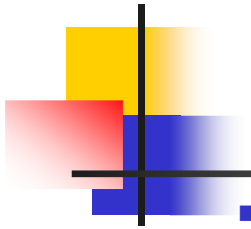
The clustered NFS feature includes:

- Monitoring:
  Every node in the NFS cluster runs an NFS monitoring utility that monitors GPFS, the NFS server and networking components on the node. After failure detection the monitoring utility may invoke a failover.
- Failover:
  The automatic failover procedure transfers the NFS serving load from the failing node to another node in the NFS cluster. The failure is managed by the GPFS cluster, including NFS server IP address failover and file system lock and state recovery.
- Load balancing:
  Load balancing is IP address based. The IP address is the load unit that can be moved from one node to another for failure or load balancing needs.
  This solution supports a failover of all the node's load as one unit to another node. However, if no locks are outstanding, individual IP addresses can be moved to other nodes for load balancing purposes.

Known GPFS limitations

- Number of filesystems < 256
- Number of storage pools < 9
- Number of cluster nodes < 4096 (2441 tested)
- Single disk (LUN) size = it is now an OS and disk limitation only
- Filesystem size < 2^{99} bytes (2 PB tested)
- Number of files < $2*10^9$
- Does not support the Red Hat EL 4.0 uniprocessor (UP) kernel.
- Does not support the RHEL 3.0 and RHEL 4.0 hugemem kernel.
- Although GPFS is a POSIX-compliant file system, some exceptions apply to this:
  - Memory mapped files are not supported in this release.
  - The *stat()* is not fully supported. *mtime*, *atime* and *ctime* returned from the *stat()* system call may be updated slowly if the file has recently been updated on another node

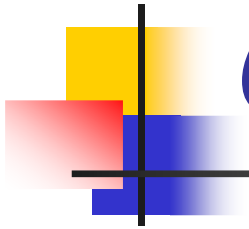
Where to Find Things in GPFS



- **Some useful GPFS directories**
  - `/usr/lpp/mmfs`
    - `/bin`... commands (binary and scripts)
      - most GPFS commands begin with "mm"
    - `/gpfsdocs`... pdf and html versions of basic GPFS documents
    - `/include`... include files for GPFS specific APIs, etc.
    - `/lib`... GPFS libraries (*e.g.*, libgpfs.a, libdmapi.a)
    - `/samples`... sample scripts, benchmark codes, etc.
  - `/var/adm/ras`
    - error logs
      - files... mmfs.log.<time stamp>.<hostname> (new log every time GPFS restarted)
      - links... mmfs.log.latest, mmfs.log.previous
  - `/tmp/mmfs`
    - used for GPFS dumps
    - sysadm must create this directory
    - see **mmconfig** and **mmchconfig**
  - `/var/mmfs`
    - GPFS configuration files
  - same directory structure for both AIX and Linux systems
- **Today's trivia question...**

Question: What does mmfs stand for?

Answer: **Multi-Media File System**... predecessor to GPFS in the research lab

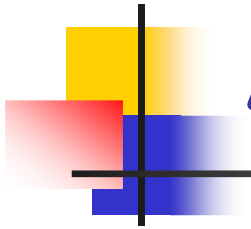


GPFS FAQs

- **Common GPFS FAQs**
  - http://publib.boulder.ibm.com/clresctr/library/gpfs_com_faq.html
- **GPFS for AIX FAQs**
  - http://publib.boulder.ibm.com/clresctr/library/gpfs_aix_faq.html
- **GPFS for Linux FAQs**
  - http://publib.boulder.ibm.com/clresctr/library/gpfs_lin_faq.html
- **COMMENT**
  - These web pages are very helpful documents on GPFS. The GPFS
  - development team keeps them relatively up to date.

Comments on Selected GPFS Manuals in

- GPFS Documentation
  - **Concepts, Planning and Installation Guide**
    - One of the most helpful manuals on GPFS... it provides an excellent conceptual overview of GPFS.
    - If this were a university class, this manual would be your assigned reading. :->
  - **Administration and Programming Reference**
    - Documents GPFS related administrative procedures and commands as well as an API guide for GPFS extensions to the POSIX API. The command reference is identical to man pages.
  - **Problem Determination Guide**
    - Many times, GPFS error messages in the mmfs.log files have an error number. You can generally find these referenced in this guide with a brief explanation regarding the cause of the message. They will often point to likely earlier error messages helping you to find the cause of the problem as opposed to its symptom.
  - **Data Management API Guide**
- Documentation available online at...
  - http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp
  - Available with the GPFS SW distribution in `/usr/lpp/mmfs/gpfsdocs`
    - note to sysadm's... Be sure to install this directory! **Also install the man pages!**
- IBM Redbooks and Redpapers
  - **http://w3.itso.ibm.com/...** do a search on GPFS



Acknowledgements

Materials used in this presentation, along with presenter's own experience, have been mainly obtained from the following sources:

- **"GPFS Programming, Configuration and Performance Perspectives"** by Raymond L. Paden, Deep Computing, IBM, 2005
- **"An Introduction to GPFS Version 3.2"** by Scott Fadden, IBM Corporation, 2007
- IBM Cluster Information Center Website: http://publib.boulder.ibm.com

.