GPFS in Today's HPC Processing Center

"This is not your father's GPFS"

Raymond L. Paden, Ph.D.
HPC Technical Architect
Deep Computing

3 June 2005

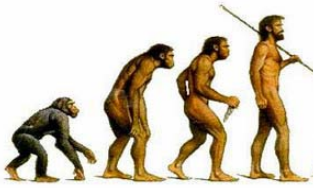
raypaden@us.ibm.com
713 - 940 - 1084

Special Notices from IBM Legal

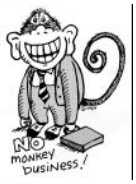
Abstract

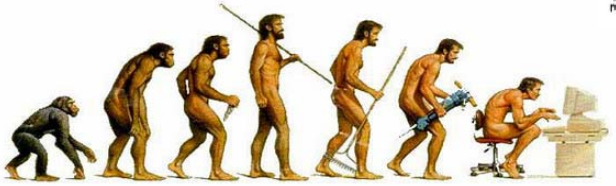
GPFS has matured significantly over the years since its version 1.x releases for AIX, yet many HPC practitioners do not fully realize GPFS's flexibility and ease of use today. This presentation examines these new features including multi-clustering, GPFS in a mixed AIX/Linux environment, its ability to work with a wider variety of disk vendors, and other things. Sample configurations with benchmark results will be given. The presentation will include a cursory review of the GPFS roadmap.

Some say people have evolved...



Some say people have evolved into something intelligent?!?!





But what about HPC systems?

How have they evolved?

A Common HPC Evolutionary Path

Mainframe
IBM mainframe with attached vector processor(s)

Vector Processor
Cray (with attached mainframe?)

A Common HPC Evolutionary Path

Mainframe
IBM mainframe with attached vector processor(s)

After n steps...

Proprietary Clusters
Room full of SPs

Vector Processor
Cray (with attached mainframe?)

A Common HPC Evolutionary Path

Mainframe
IBM mainframe with attached vector processor(s)

Vector Processor
Cray (with attached mainframe?)

After n steps...

Then some renegade starts experimenting with Beowulf clusters...

Proprietary Clusters
Room full of SPs

starting like this...

A Common HPC Evolutionary Path

Mainframe
IBM mainframe with attached vector processor(s)

Vector Processor
Cray (with attached mainframe?)

After n steps...

Then some renegade starts experimenting with Beowulf clusters...

Proprietary Clusters
Room full of SPs

starting like this...

then evolving into this...

A Common HPC Evolutionary Path

Mainframe
IBM mainframe with attached vector processor(s)

Then some renegade starts experimenting with Beowulf clusters...

Beowulf cluster

Rack mounted Linux Nodes
IBM Cluster 1350

After n steps...

Proprietary Clusters
Room full of SPs

Blades (using Linux)
IBM BladeCenter

Vector Processor
Cray (with attached mainframe?)

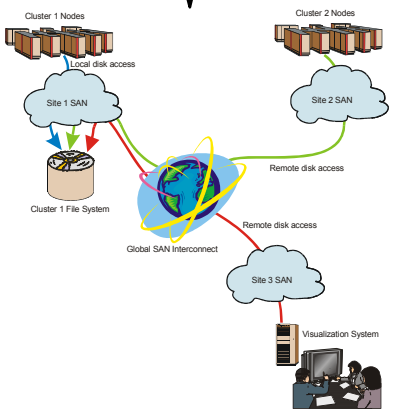
Proprietary SMP Clusters
Room full of IBM p690s

What Next?

Proprietary SMP Clusters
Room full of IBM p690s

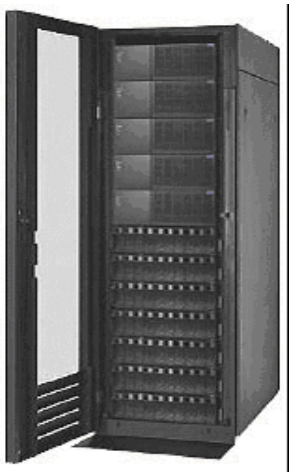
Rack mounted Linux Nodes
IBM Cluster 1350

Blades (using Linux)
IBM BladeCenter



... and everybody is talking about grids!

But Where Does Storage I/O Fit?



Storage I/O...
The oft forgotten stepchild

But Where Does Storage I/O Fit?

- **Early adopters of proprietary clusters (e.g., IBM SP) generally adopted vendor storage solutions (e.g., SSA and GPFS or JFS)**
  - GPFS is NOT the same beast it used to be!
- **Early adopters of commodity clusters approached storage I/O from a potpouri of approaches (e.g., NFS)**
  - There are alternatives to NFS
- **Customers trying to integrate proprietary and commodity systems often feel forced to use NFS**
  - There are alternatives to NFS
- **And what about grids?**

Lets take a closer look at this.

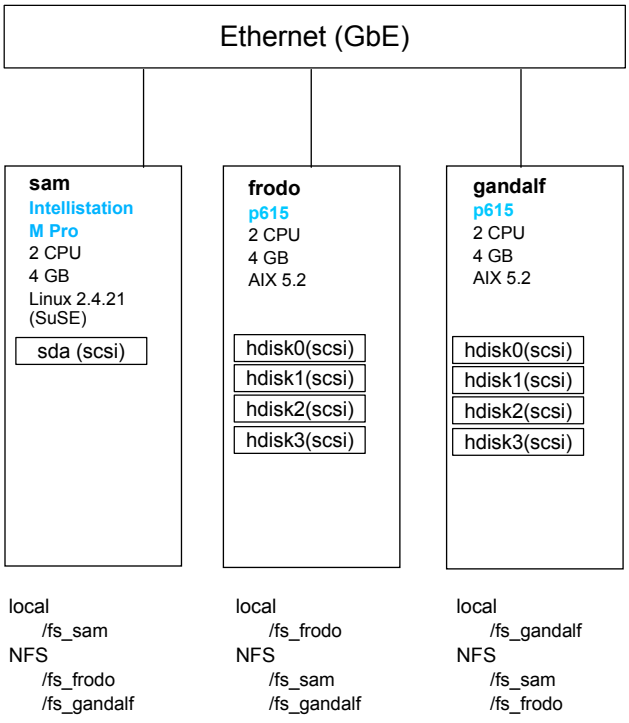
I will begin with the Linux clusters perspective.

I will get to the SP to pSeries perspective in a moment.

He who does not study history is pre-destined to relive it... errr, but is NFS really history?

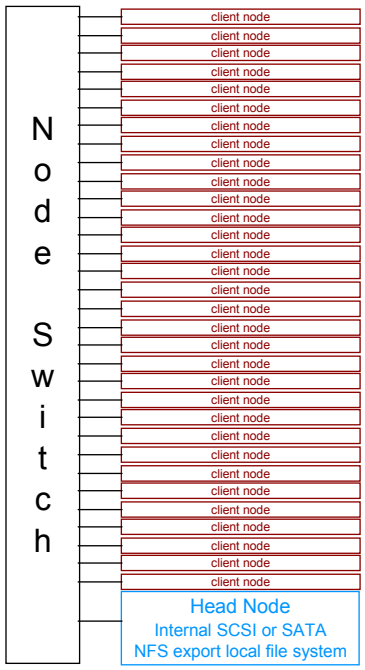
Common First Step
For Something Small

For something like this, it is common to do NFS and FTP between servers over a GbE or 100 MbE network.



Common Second Step

Make the Small Solution BIGGER



"Client" Nodes

- e.g., x336 (Linux)
- access to NFS mounted file system from head node
- internal SCSI used for local scratch
  - FTP files as necessary between clients

Head Node

- e.g., x346 (Linux)
- file system based on internal storage and NFS exported to client

Node Switch

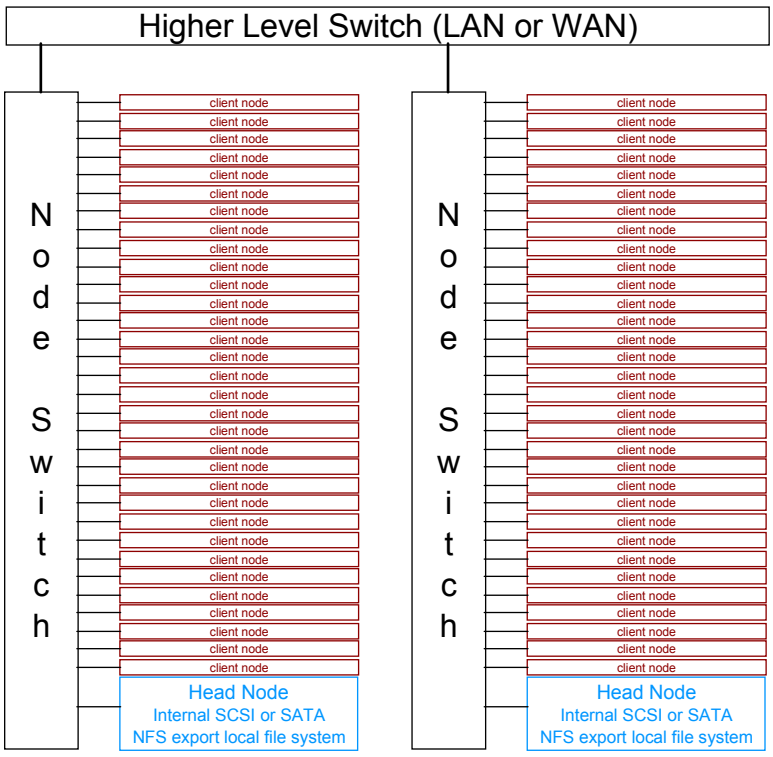
- Generally, IP based network
  - GbE
  - Myrinet
  - HPS
  - IB?

Common Application Organization

- Use IP network to distribute data via MPI, NFS and/or other "home-grown mid-layer" codes
  - works well for applications using minimal or no parallel I/O
  - do application developers (i.e., computational scientists) want to become computer scientists?

Common Third Step

Create "Islands of Nodes" as You Get Even BIGGER



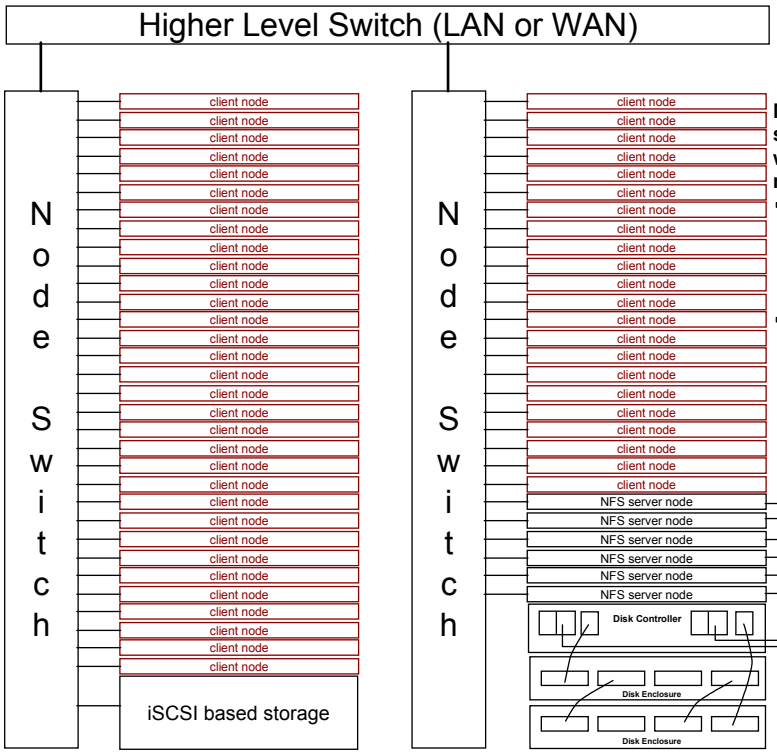
Clusters are connected via a heirarchical switching network

COMMENTS

- Provides "any to any" connectivity
  - the poor man's way
- Works well when I/O model is not parallel, but may require aggregating files
- ISLs can be bottlenecks
- Inadequacy of NFS semantics (especially for parallel writes)
- Poor I/O performance
- Limited storage capacity
  - can add more head nodes
- Storage BW limited by headnode switch adapter
- Inconvenience of ftp or other utilities to manually move files
- Common fabric for message passing and storage I/O
- Naturally generalizes to a grid with all of its issues, but is compounded by varibilities of geographic seperation!

Common Third Step

Create "Islands of Nodes" as You Get Even BIGGER



Clusters are connected via a heirarchical switching network

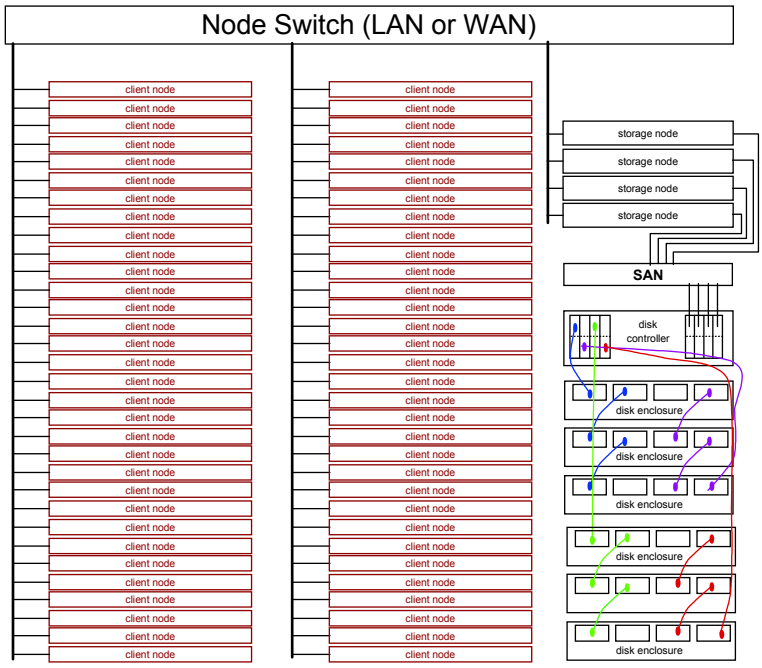
More sophisticated storage systems can be adopted to work within this NFS/IP based model over a WAN/grid

- iSCSI based systems
  - NFS not necessarily required
  - "plain vanilla" iSCSI can be used, but more sophisticated schemes are being investigated (e.g., file replication, Univ of Tokyo)
- SAN based file system provides local I/O but the local file systems are NFS exported over the WAN
  - still must deal with NFS shortcommings

In this SAN example, the SAN attached storage would typically be distributed over the 6 servers with 6 or more different NFS exported file systems (at least 1 file system per NFS server).

The Final(?) Step

Global Storage with Parallel File System



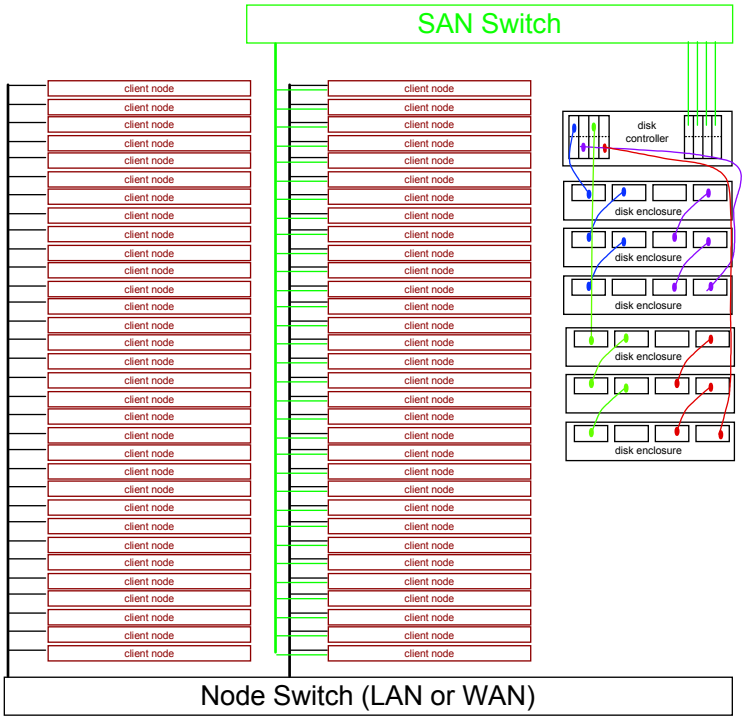
Storage servers with external storage using the node switch

COMMENTS

- homogenous network
- parallel FS fascilitates the effective use of this architecture
- disks are accessed via the LAN/WAN
  - virtual disks
- performance scales linearly in the number of servers
  - increasing the number of servers will increase BW
- can add capacity without increasing the number of servers
- server switch adapters can become a bottleneck
- can inexpensively scale out the number of nodes
  - largest GPFS cluster with a single file system 2300 nodes
- natural model for grid based file system

The Final(?) Step

Another Global Storage Parallel File System Model



SAN attached storage

COMMENTS

- seperate switch fabrics
- parallel FS fascilitates the effective use of this architecture
- performace scales in the number of disk controllers
- can add capacity without increasing the number controllers
- scaling out the number of direct attach nodes is limited by the SAN
  - largest SAN cluster is 200+ nodes
  - scaling larger requires remote nodes accessing storage over IP network
  - direct attach nodes get better file system BW
- BW is not restricted by server node switch adaptes (typically, a FC HBA is faster the GbE... but does this change with IB?)
- Allows greater aggregate BW
  - e.g., 15 GB/s on 40 nodes
- SAN works well in a processing center
- Use a LAN/WAN to scale out beyond SAN limits

In the early days of HPC clusters, there were limited choices for parallel/global file systems... and generally it was necessary to use the vendor's file system. Today there are other choices (at least 18 at my last count) that have been enabled by the development of Linux based clusters.

In order to more clearly understand how GPFS fits into this environment, the following pages discuss a coarse HPC storage architecture taxonomy covering range of file systems used on an HPC systems... this is a work in progress!

HPC Storage Architecture Taxonomy

The following pages examine an architectural taxonomy of storage I/O architectures commonly used in HPC systems. They support varying 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
- **Recent Developments**
  - Triple Component Architecture
- **High Level Parallel I/O**

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)
  - more favorable toward temporal than spatial locality
- Intra-node process parallelism
- Disk level parallelism possible via striping
- Not truly a parallel file system
- Examples: Ext3, JFS2, XFS

COMMENT: GPFS cache (i.e., pagepool) is more favorable toward spatial than temporal locality. Very large pagepools (up to 8 GB using 64 bit OS) may do better with temporal locality.

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 supports AIO

Networked File System (NFS)

- Disk access from remote nodes via network access (*e.g.*, TCP/IP over Ethernet)
- NFS is ubiquitous and the most common example
  - it is not truly parallel
    - old versions are not cache coherent (is V3 or V4 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)

COMMENT: enhancements have been proposed for NFS V4 under AIX that should improve NFS parallel writes.

  - 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 customers get "good enough" performance from it

Basic Parallel I/O
Single Component Architecture

- Parallelizes file, metadata and control operations
  - single component architecture: does not require distinction between metadata, storage and client nodes
- 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 applications using 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 (IBM...best of class), GFS (Sistina/Redhat)

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
- Dual component architecture (storage client/server, metadata server)
- 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 (but SANs already limit scaling). Latency more than BW is potential issue.
- 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 (and can go as high as 100 nodes), and is capable of very high BW
    - on a very carefully configured/tuned p690, GPFS, JFS2 and SNFS all got 15 GB/s
  - CXFS scales only to 10-12 servers for some users, perhaps at most 30?
    - good enough for large SMPs?
- Example: CXFS (SGI), SNFS (ADIC), SanFS (IBM)
  - SanFS and SNFS place heavy emphasis on storage virtualization

Recent Developments
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 taxonomy. They have a number of architectural similarities.

- **Triple component architecture**
  - storage clients, storage servers, metadata servers
  - 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 fascilitate storage management (i.e., "storage virtualization")**
- **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 ubiquitous 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 adavantages of being a standard (e.g., syntactic portability)

Which Architecture is Best?

There is no concise answer to this question.

- It is application/customer 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 Others Say About GPFS

Two recent papers comparing/contrasting parallel file systems.

- Margo, Kovatch, Andrews, Banister. "An analysis of State-of-the-Art Parallel File Systems for Linux.", 5th International Conf on Linux Clusters: HPC Revolution 2004, Austin, TX, May 2004
  - Compared GPFS, Lustre, PVFS
  - Criteria: performance, system administration, redundancy, special features
  - "In both SAN and NSD modes, GPFS performed the best. It was also easy to install and had numerous redundancy and special features."

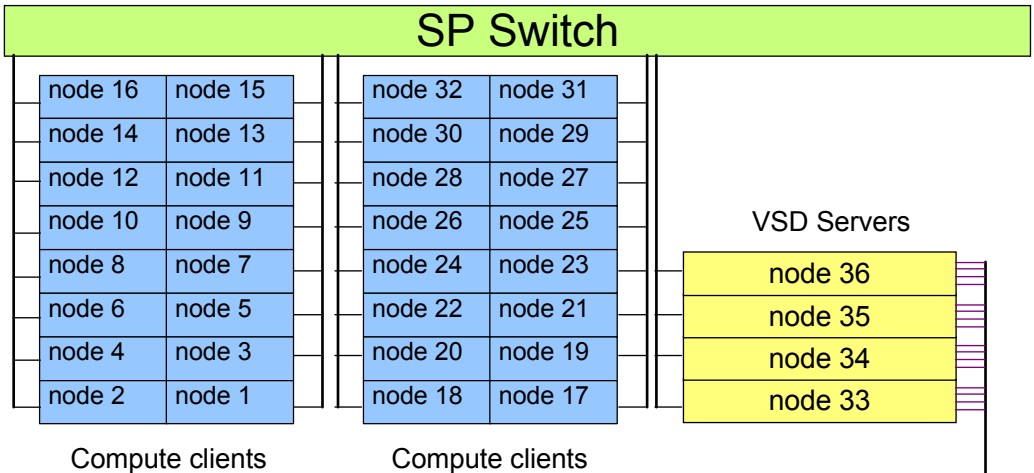
- Cope, Oberg, Tufo, Woitaszek. "Shared Parallel Filesystems in Heterogeneous Linux Multi-Cluster Environments.", 6th International Conf on Linux Clusters: HPC Revolution 2005, Chapel Hill, NC, April 2005
  - Compared GPFS, Lustre, NFS, PVFS2, TerraFS
  - Criteria: performance, usability, stability, special features
  - "Our experiences with GPFS were very positive, and we found it to be a very powerful filesystem with well documented administrative tools."

What is GPFS?

This question no longer has simple answer.

GPFS in The "Good Old Days"

A Typical "Winterhawk" Config with GPFS 1.3



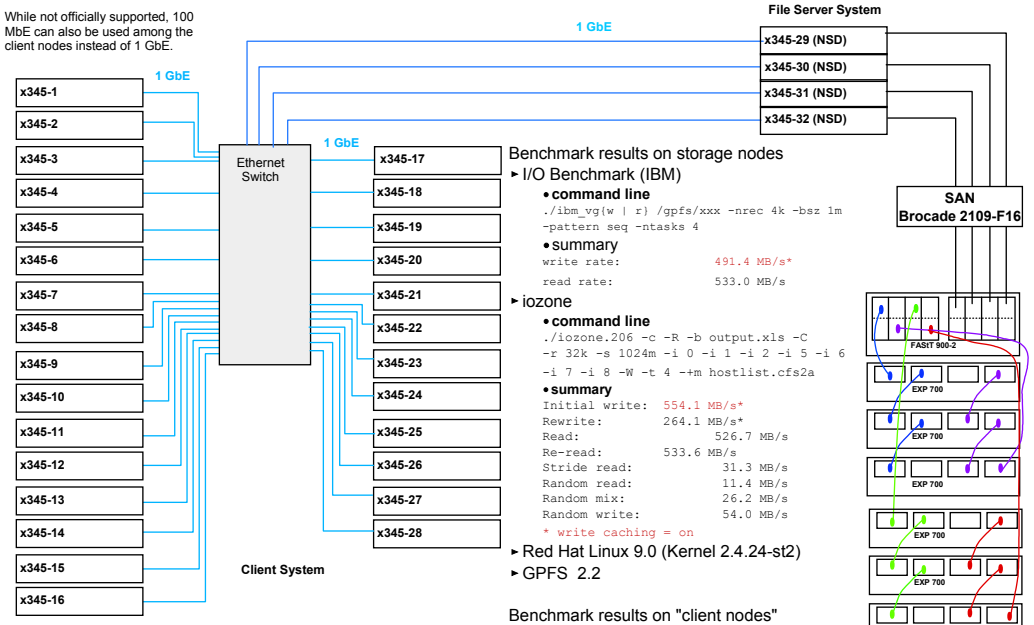
- All nodes run AIX
- Thin nodes are compute clients, wide nodes are VSD servers
- GPFS packets transit the switch
- The disk is SSA
- Peak Aggregate BW for this configuration
  - at most 440 MB/s
- Took experienced sysadm a day or 2 to configure

... but GPFS can't do that!?!?!

Old ideas die hard. GPFS is far more versitile than what it was in its early days. The following pages highlight many of these newer features.

GPFS Today

GPFS under Linux with GbE



Benchmark results on storage nodes

- I/O Benchmark (IBM)
  - **command line**

```
./ibm_vg{w | r} /gpfs/xxx -nrec 4k -bsz 1m
-pattern seq -ntasks 4
```

  - summary

```
write rate:              491.4 MB/s*
read rate:               533.0 MB/s
```

- iozone
  - **command line**

```
./iozone.206 -c -R -b output.xls -C
-r 32k -s 1024m -i 0 -i 1 -i 2 -i 5 -i 6
-i 7 -i 8 -W -t 4 -+m hostlist.cfs2a
```

  - **summary**

```
Initial write:  554.1 MB/s*
Rewrite:        264.1 MB/s*
Read:                526.7 MB/s
Re-read:        533.6 MB/s
Stride read:          31.3 MB/s
Random read:          11.4 MB/s
Random mix:           26.2 MB/s
Random write:         54.0 MB/s
* write caching = on
```

- Red Hat Linux 9.0 (Kernel 2.4.24-st2)
- GPFS 2.2

Benchmark results on "client nodes"

- I/O Benchmark (IBM), write caching **off**
- BW constrained by single GbE adapter on each NSD server
- 1 client
  - write = 92.3 MB/s, read = 111 MB/s
- 8 clients
  - write = 360 MB/s, read = 384 MB/s

Using Myrinet on 8 clients and 2 FAStT900s

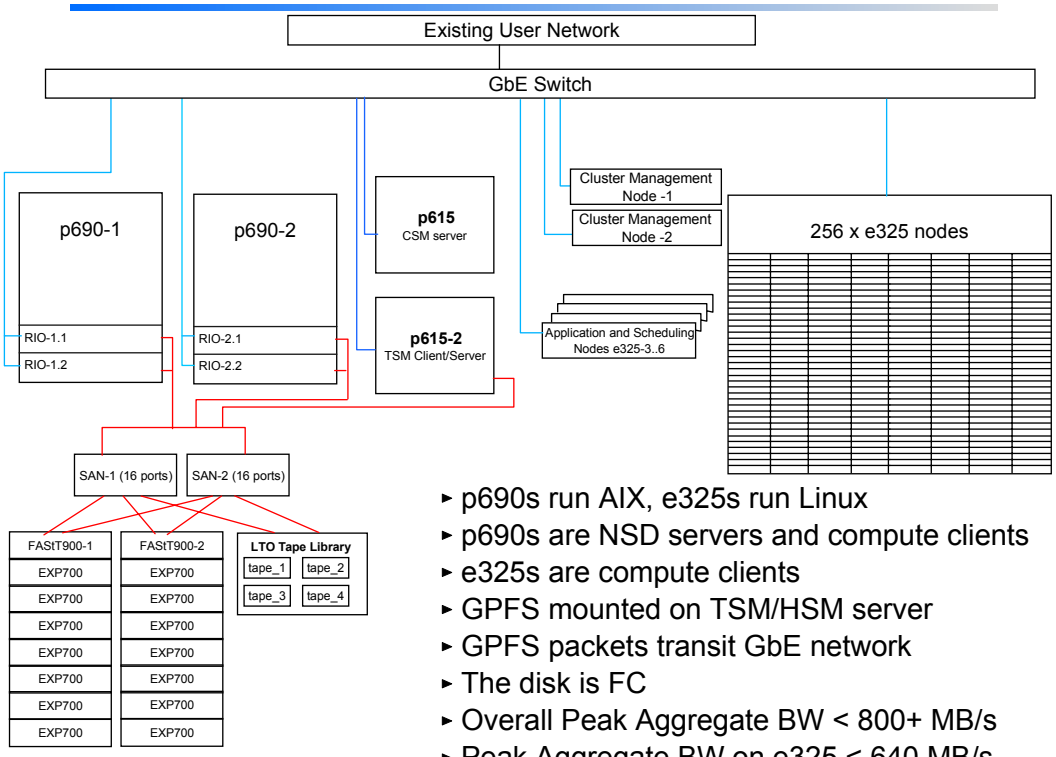
- write = 397 MB/s
- read = 585 MB/s

A second FAStT900 was needed since peak read BW exceeded the ability of 1 FAStT900.

Scaling Out:

- Actual GbE based designes have been extended upto 1100 Linux nodes (e.g., Intel or Opteron) with GPFS 2.2
- Current designed maximum for GPFS 2.3: 2000+ nodes

GPFS Today
Mixed AIX/Linux Config



- p690s run AIX, e325s run Linux
- p690s are NSD servers and compute clients
- e325s are compute clients
- GPFS mounted on TSM/HSM server
- GPFS packets transit GbE network
- The disk is FC
- Overall Peak Aggregate BW < 800+ MB/s
- Peak Aggregate BW on e325 < 640 MB/s

GPFS Today
Mixed AIX/Linux Config

Benchmark Config
- 1 p690
- 32 x335s

		Natural Aggregate		Harmonic Aggregate		Harmonic Mean		
Number of Nodes	Number of tasks per node	Write Rate MB/s	Read Rate MB/s	Write Rate MB/s	Read Rate MB/s	write	read	sizeof(file) GB
1	1	640.10	765.28					4
1	2	641.07	760.75	648.02	761.71	324.01	380.85	8
1	4	631.84	739.31	646.62	758.13	161.65	189.53	16
1	8	649.14	724.88	670.83	755.63	83.85	94.45	32
1	16	646.97	721.26	671.29	788.48	41.96	49.28	64

p690; write caching = off, pattern = sequential, bsz = 1 MB

		Natural Aggregate		Harmonic Aggregate		Harmonic Mean		
Number of Nodes	Number of tasks per node	Write Rate MB/s	Read Rate MB/s	Write Rate MB/s	Read Rate MB/s	write	read	sizeof(file) GB
1	1	109.27	110.89					4
2	1	198.92	218.65	198.92	219.51	99.46	109.76	8
4	1	269.91	435.37	269.95	437.68	67.49	109.42	16
8	1	282.44	624.19	283.44	626.81	35.43	78.35	32
16	1	253.23	595.50	281.78	598.18	17.61	37.39	64
32	1	269.77	577.53	269.83	581.23	8.43	18.16	128

x335; write caching = off, pattern = sequential, bsz = 1 MB

			Natural Aggregate		Harmonic Aggregate		Harmonic aggregate over x335 and p690		
	Number of Nodes	Number of tasks per node	Write Rate MB/s	Read Rate MB/s	Write Rate MB/s	Read Rate MB/s	write	read	sizeof(file) GB
x335 only ***	4	1	267.33	435.72	268.29	437.23	N/A	N/A	16
p690 only	1	4	632.45	771.00	650.98	788.25	N/A	N/A	16
x335 with p690	4	1	233.03	386.27	233.03	389.61	707.48*	801.66*	24**
p690 with x335	1	4	470.22	407.79	477.30	412.81			32**

mixed nodes; write caching = off, pattern = sequential, bsz = 1 MB, ntasks = 4

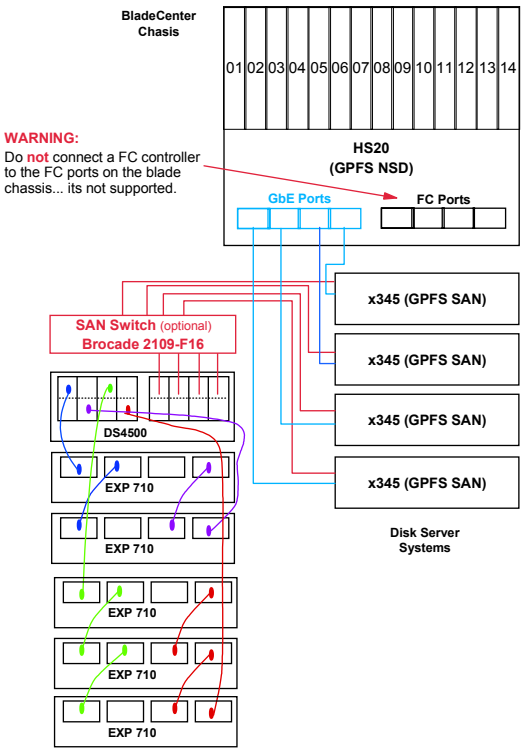
* Job times are nerely identical; therefore, iostat measured rate was very close to harmonic aggregate rate.

** Size of combined files from each job; file sizes adjusted so job times were approximately equal.
Combined files for write = 24 GB, combined files for the read = 32 GB.

*** x335 aggregate read rates were gated by the 4 GbE at a little over 100 MB/s per adapter.

GPFS Today

GPFS in a BladeCenter - Standard Configuration



SYSTEM ANALYSIS

1. DS4500
   - sustained peak performance < **540 MB/s**
2. **FC Network**
   - sustained peak performance < **600 MB/s**
3. **GbE Network** (adapter aggregate measured over all 4 x345s)
   - sustained peak performance < **360 MB/s**
4. **Aggregate x345 Rate**
   - sustained peak performance < **500 MB/s**
5. **Predicted Aggregate HS20 Rate**
   - sustained peak performance < 360 MB/s

Comments
- HS20 performance constrained by limited GbE Ports
- The rates for items 1-4 are based on benchmark tests
- The SAN switch is optional; using it *may* reduce load on GbE network and reduce aggregate application disk I/O bandwidth

Lower Cost/Bandwidth Alternative

If less file access bandwidth is required or a lower cost solution is required, then the x345/FAStT900 system can be replaced with the following:
- 2 disk servers (x345) each with 1 GbE and 1 FC HBA
- 1 FAStT600 and 1 disk enclosure (EXP700)
- SAN switch is optional
- sustained peak performance < 200 MB/s

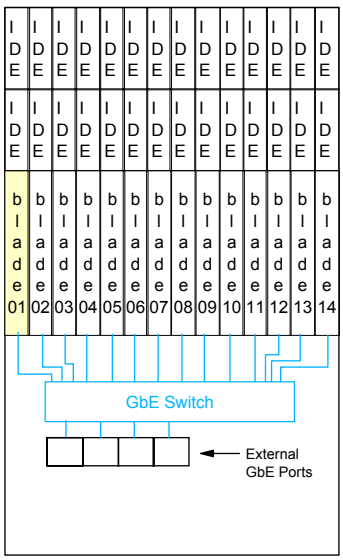
Global File System over Multiple HS20 Systems

This stand alone system can be replicated N times. By routing HS20 and x345 GbE traffic through a switch, the NSD layer in GPFS will enable all blades to see all LUNs; i.e., multiple HS20 systems can all safely mount the same GPFS file system and performance will scale linearly.

GPFS Today

GPFS in a BladeCenter - Alternative Configuration

Blade Center
- Internal GbE network highlighted in blue



NOTE:

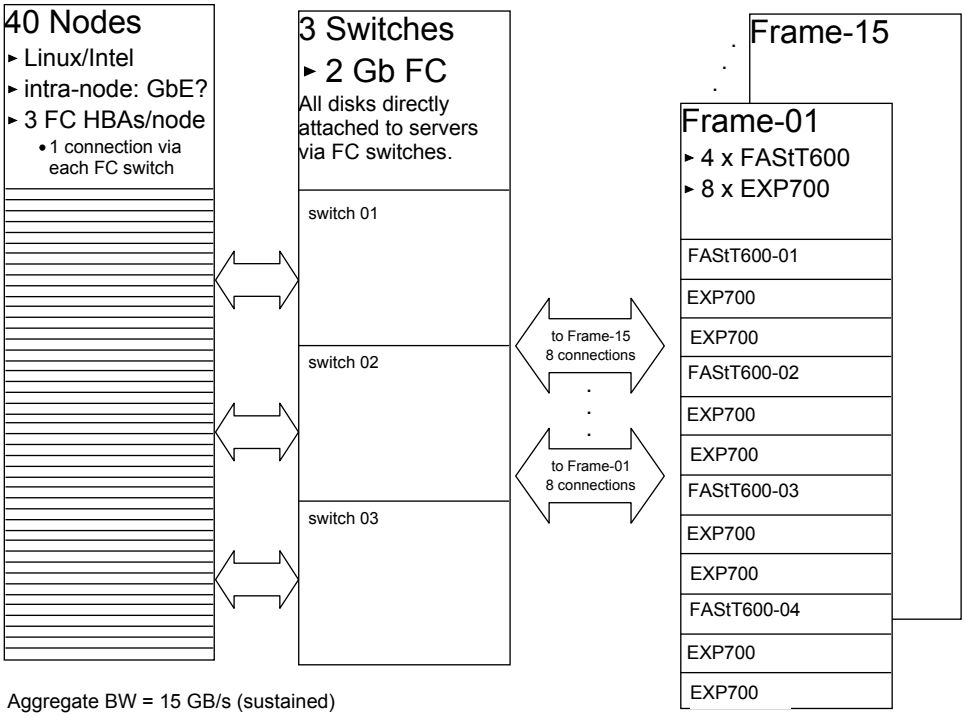
This is not recomended configuration by GPFS since the disks are not twin tailed... "but it works".

BENCHMARK ANALYSIS

- A private internal GbE network connects all blades
- Each blade has a single GbE adapter
  - effective BW = 80 to 100 MB/s
- Baseline performance (read from a single local disk using ext2)
  - application read rate = **30 MB/s**
- Single task GPFS performance
  - application read rate = **80 MB/s**
  - 2.7 X faster than single disk rate
- Further analysis
  - assume active task is on blade 01
  - GPFS stripes over all 28 IDE drives in this configuration
  - GPFS uses GbE network for striping activity, therefore single blade GPFS performance limited to GbE adapter BW (i.e., upto 100 MB/s)
  - each blade
    - has only one GbE adapter used by GPFS and general system
    - acts as a disk server and GPFS client
    - single task performance = **80 MB/s**
  - in a similar test on x345 where
    - there were seperate GPFS clients and disk servers
    - there were 2 GbE adapters per node (one for GPFS and one for everything else)
    - single task performance = **100 MB/s**
- Aggregate rate (1 task per blade)
  - read rate = **560 MB/s**
- Analysis
  - each blade is acting as a disk server (as well as client)
  - since GPFS scales linearly in the number disk servers, it yields good aggregate performance
  - in a Winterhawk 2 system using SSA disk with each node acting as a GPFS client and disk server, the aggregate rate over 14 nodes was **< 420 MB/s**

GPFS Today

SDSC/IBM StorCloud Architecture



Aggregate BW = 15 GB/s (sustained)
goto: http://www.sdsc.edu/Press/2004/11/111204_SC04.html

GPFS Today

Selected SDSC/IBM StorCloud Statistics

- **40 Linux Nodes**
- **3 FC HBAs per Node**
- **15 Storage frames**
- **60 FAStT600s**
- **2520 disks**
- **240 LUNs**
- **8+P**
- **4 LUNs per FAStT600**
- **73 GB/disk @ 15 Krpm**
- **Sustained Aggregate Rate**
  - 15 GB/s
  - 380 MB/s per node
  - 256 MB/s per FAStT600

COMMENT: IP vs FC

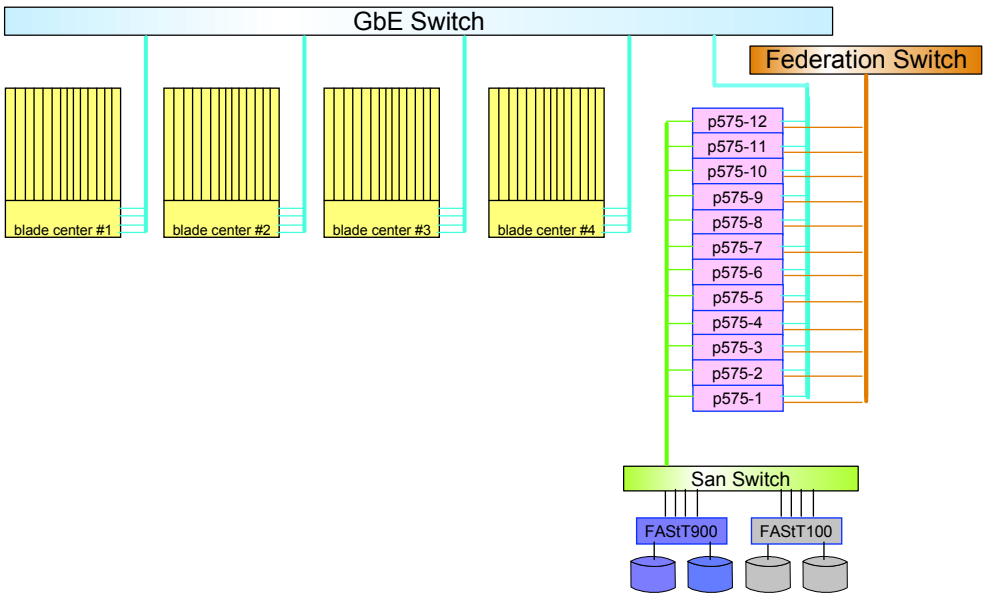
With today's technology, direct attached disk models (e.g., SAN attached) can yielder greater per node BW than IP based models.

- IP based systems are limited by ethernet adapter (e.g., 80 MB/s for 1 GbE, 120 MB/s for dual bonded GbE)
- direct attached systems can have multiple FC HBAs (e.g., with 3 HBA/node the BW is 380 MB/s)
- Will 10 GbE change this?
- Will IB change this?

GPFS Today
Storage Pools

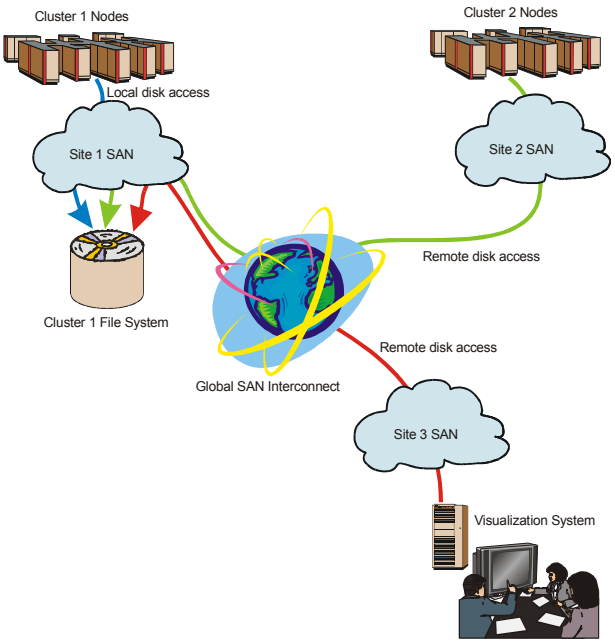
- **Motivation**
  - Some newer file systems implement a concept called "storage pools"; GPFS supports a form of this.
- **Disks present themselves to GPFS as LUNs**
- **A GPFS file system can mount a FS on any subset of these LUNs**
  - There is 1 storage pool per FS
  - Max: 32 file systems per GPFS cluster
- **Example**
  - Monolythic disk architecture (e.g., SATA)
  - Access is "bursty"
  - To avoid striping over all disks and stressing all disks, divide disks into 16 disjoint subsets and hence 16 file systems. File striping is confined to a file system. When a file system not in use, GPFS is not spinning disks. (n.b., All FS's are seen by all nodes in the GPFS cluster... upto 2000+ nodes)
- **Example**
  - 2 classes of disk: FC disk and SATA disk
  - 1 FS for FC disk used for constant access
  - 1 FS for SATA disk used infrequently

GPFS Today
Storage Pools in a Mixed BladeCenter/pSeries Cluster

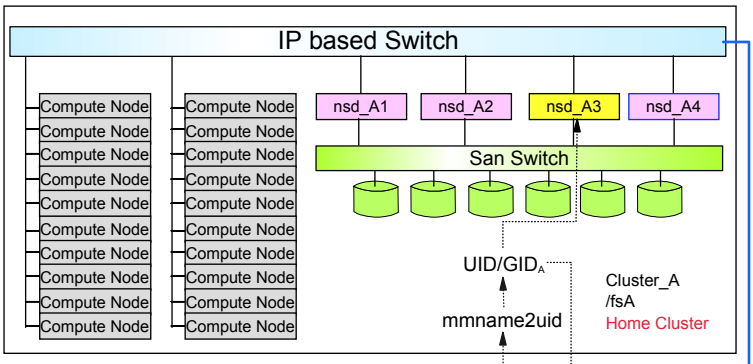


GPFS Today
Cross-cluster Mounts

- Problem: nodes outside the cluster need access to GPFS files
- Solution: allow nodes outside the cluster to mount the file system
  - "Owning" cluster responsible for admin, managing locking, recovery, …
  - Separately administered remote nodes have limited status
    - Can request locks and other metadata operations
    - Can do I/O to file system disks over global SAN (IP, Fibre Channel, …)
    - Are trusted to enforce access control, map user Ids, …
- Uses:
  - High-speed data ingestion, postprocessing (e.g. visualization)
  - Sharing data among clusters
  - Separate data and compute sites (Grid)
  - Forming multiple clusters into a "supercluster" for grand challenge problems



GPFS Today
Cross-cluster Mounts -- Example

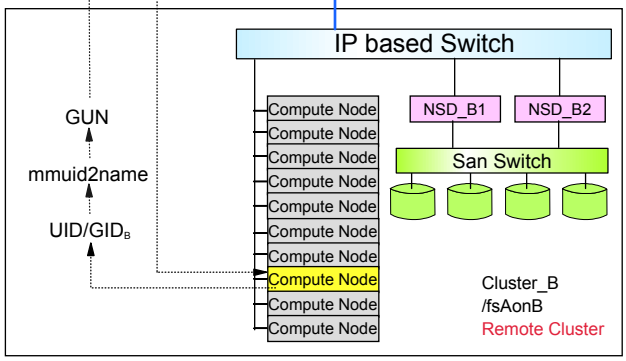


Inter-Switch Links (at least GbE speed!)

COMMENTS:
- Cluster_B accesses /fsA from Cluster_A via the NSD nodes
  - see example on next page
- Cluster_B mounts /fsA locally as /fsAonB
- OpenSSL (secure socket layer) provides secure access between clusters

UID MAPPING EXAMPLE (i.e., Credential Mapping)

1. pass Cluster_B UID/GID(s) from I/O thread node to mmuid2name
2. map UID to GUN(s) (Globally Unique Name)
3. send GUN(s) to mmname2uid on node in Cluster_A
4. generate corresponding CLUSTER_A UID/GID(s)
5. send Cluster_A UID/GIDs back to Cluster_B node runing I/O thread (for duration of I/O request)



COMMENTS:
- mmuid2name and mmname2uid are user written scripts made available to *all* users in /var/mmfs/etc; these scripts are called *ID remapping helper functions* (IRHF) and implement access policies
- simple strategies (e.g, text based file with UID <-> GUN mappings) or 3rd party packages (e.g., Globus Security Infrastruction from Teragrid) can be used to implement the remapping procedures

See http://www-1.ibm.com/servers/eserver/clusters/whitepapers/uid_gpfs.html for details.

GPFS Today
Cross-cluster Mounts -- Sysadm Commands

Mount a GPFS file system from Cluster_A onto Cluster_B

- assume diagram from the previous page

On Cluster_A

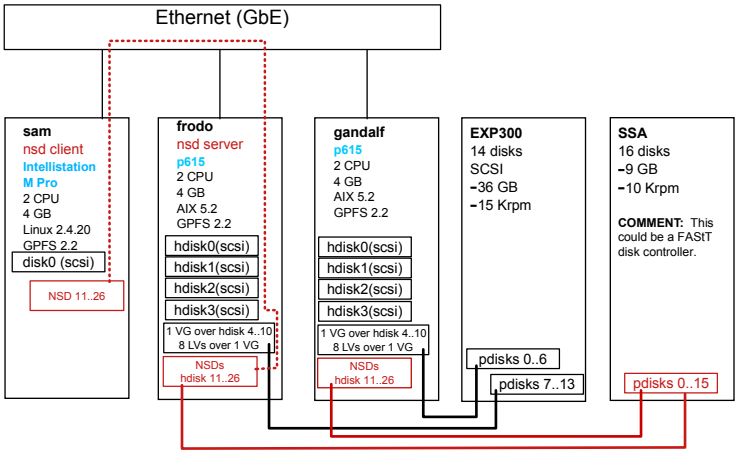
1. Generate public/private key pair
   `mmauth genkey`
   COMMENTS
   - creates public key file with default name `id_rsa.pub`
   - start GPFS daemons *after* this command!
2. Enable authorization
   `mmchconfig cipherList=AUTHONLY`
3. Sysadm gives following file to Cluster_B
   `/var/mmfs/ssl/id_rsa.pub`
   COMMET: rename as `cluster_A.pub`
7. Authorize Cluster_B to mount FS owned by Cluster_A
   `mmauth add cluster_B -k cluster_B.pub`

On Cluster_B

4. Generate public/private key pair
   `mmauth genkey`
   COMMENTS
   - creates public key file with default name `id_rsa.pub`
   - start GPFS daemons *after* this command!
5. Enable authorization
   `mmchconfig cipherList=AUTHONLY`
6. Sysadm gives following file to Cluster_A
   `/var/mmfs/ssl/id_rsa.pub`
   COMMENT: rename as `cluster_B.pub`
8. Define cluster name, contact nodes and public key for cluster_A
   `mmremotecluster add cluster_A -n nsd_A1,nsd_A2,nsd_A3,nsd_A4 -k Cluster_A.pub`
9. Identify the FS to be accessed on cluster_A
   `mmremotefs add /dev/fsAonB -f /dev/fsA -C Cluster_A -T /dev/fsAonB`
10. mount FS locally
    `mount /fsAonB`

See http://publib.boulder.ibm.com/clresctr/docs/gpfs/gpfs23/200412/bl1adm10/bl1adm1031.html#admmcch for details.

GPFS Today
GPFS is Easier to Administer



To build the file system*, do the following on gandalf...

1. mmcrcluster
2. mmstartup
3. mmcrnsd
   - specify primary, secondary, client, server nodes in disk descriptor file
4. mmcrfs
5. mount /<FS name>

* GPFS 2.3

COMMENTS:

- Once the SAN zoning and low level disk formats are complete, one can build GPFS in under 5 minutes on smaller systems.
- For StorCloud, it took ~= 30 minutes, but this was over a 135 TB file system (n.b., 240 LUNs or 2520 disks)
- Other dymanic features...
  - mmadddisk, mmaddnode, mmdeldisk, mmdelnode
  - mmchattr, mmchfs, mmchcluster, mmchconfig, mmchnsd
  - mmpmon

So what is GPFS?

... in one line or less

So what is GPFS?

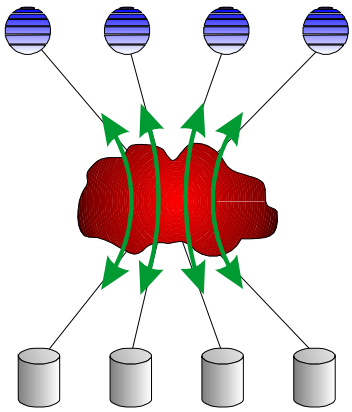
It is IBM’s shared disk, parallel file system for AIX and Linux clusters.

What is GPFS?

GPFS = General Parallel File System

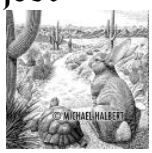
IBM's shared disk, parallel file system for AIX, Linux clusters

- *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

1. General Parallel File System
   - mature IBM product generally available for 7 years
2. Clustered, shared disk, parallel file system for AIX and Linux
3. Adaptable to many customer environmnets by supporting a wide range of basic configurations and disk technologies
4. Provides safe, high BW access using the POSIX I/O API
5. Provides non-POSIX advanced features
   - e.g., DMAPI, data-shipping, multiple access hints (also used by MPI-IO)
6. Provides good performance for large volume, I/O intensive jobs
7. Works best for large record, sequential access patterns, has optimizations for other patterns (*e.g.*, strided, backward)
8. Strong RAS features (reliability, accessibility, serviceability)
9. Converting to GPFS does not require application code changes provided the code works in a POSIX compatible environment



GPFS Features

GPFS Performance Features

1. striping
2. large blocks (with support for sub-blocks)
3. byte range locking (rather than file or extent locking)
4. access pattern optimizations
5. file caching (*i.e.*, pagepool) that optimizes streaming access
6. prefetch, write-behind
7. multi-threading
8. distributed management functions (*e.g.*, metadata, tokens)
9. 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

1. If a node fails providing GPFS management functions, an alternative node assumes responsibility reducing risk of loosing the file system.
2. 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
3. In SAN environment, failover reduces risk of lost access to data.
4. GPFS on RAID architectures reduces risk of lost data.
5. Online and dynamic system management allows file system modifications without bringing down the file system.
   - mmadddisk, mmdeldisk, mmaddnode, mmdelnode, mmchconfig, mmchfs

GPFS Features

Other Features

1. Disk scaling allowing large, single instantiation global file systems (100's of TB now, PB in future)
2. Node scaling (2300+ nodes) allowing large clusters and high BW (many GB/s)
3. Multi-cluster architecture (i.e., grid)
4. Journaling (logging) File System - logs information about operations performed on the file system meta-data as atomic transactions that can be replayed
5. Data Management API (DMAPI) - Industry-standard interface allows third-party applications (e.g. TSM) to implement hierarchical storage management

Why is GPFS needed?

Clustered applications impose new requirements on the file system

- "Any to any" access
  - any node in the cluster has access to any data in the cluster
- Parallel applications need access to the same data from multiple nodes
- Serial applications dynamically assigned to processors based on load
  - need high-performance access to their data from wherever they run
- Require both good availability of data and normal file system semantics
- Scalability to large numbers of nodes

GPFS supports this via:

- Uniform access – single-system image across cluster
- Conventional Posix API – no program modification
- High capacity – large files, 100TB + file system
- High throughput – wide striping, large blocks, many GB/sec to one file
- Parallel data and metadata access – shared disk and distributed locking
- Reliability and fault-tolerance - node and disk failures
- Online system management – dynamic configuration and monitoring

Example GPFS Applications

- Customer applications that require fast, scalable access to large amounts of file data. These applications may be serial or parallel, reading or writing.
  - Applications that serve data to visualization engines
  - Seismic data acquisition processing for serial or parallel reading/writing of files
- Environments with very large data, especially when single file servers (such as NFS) reach capacity limits
  - Digital library file serving
  - Access to large CAD/CAM file sets
  - Data mining applications
  - Data cleansing applications preparing data for data warehouses
  - Oracle RAC
- Applications requiring data rates which exceed what can be delivered by other file systems
  - Large aggregate scratch space for commercial or scientific applications
  - Internet serving of content to users with balanced performance
- Applications with high availability (HA) file system requirements

Selected New Features in GPFS 2.3

- **Scale out to larger clusters (over 2000)**
  - Current largest production cluster is 2300 blades
- **Bigger file systems**
  - 100's of TB
- **Bigger LUNs**
  - There is no GPFS limitation; it is now an OS and disk limitation only
    - over 2 TB on AIX in 64 bit mode, up to 2 TB in "other supported" OS's
- **Depends less on disk protocols for many features (e.g., SCSI persistent reserve)**
  - therefore GPFS is portable to a wider variety of disk hardware
- **No longer requires RSCT**
- **Only one cluster type (n.b., no need to specify sp, rpd or hacmp)**
- **Simpler quorum definition rules**
- **GPFS specific performance monitor**
  - Measures latency and *bandwidth*
- **GPFS is easier to administer and use**