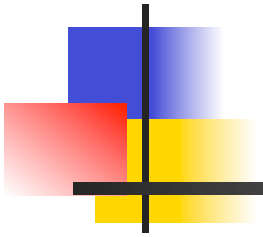
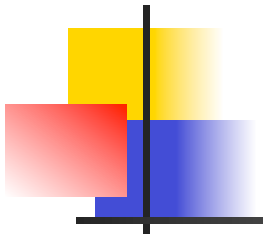


GPFS for advanced users

Part 1: General Introduction

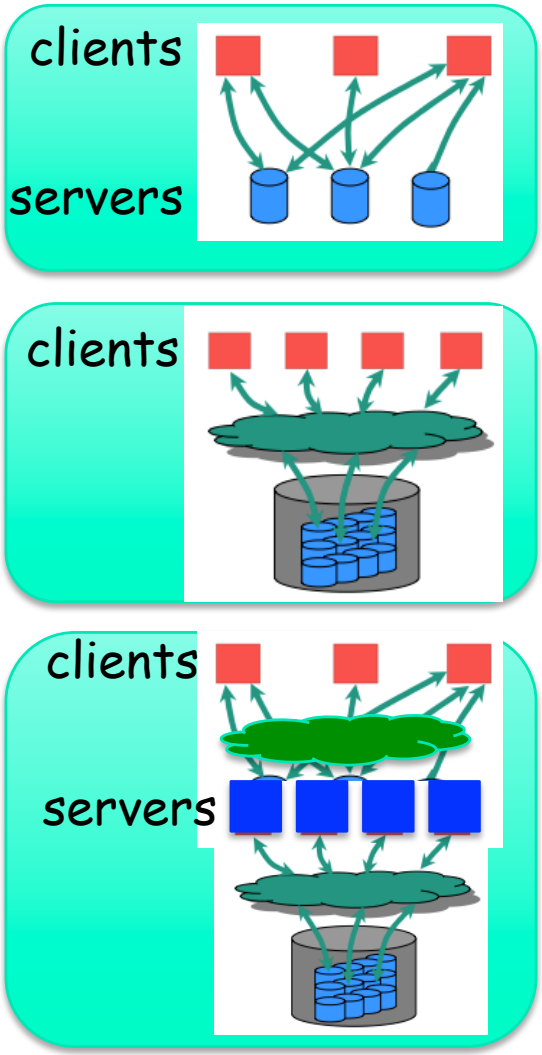


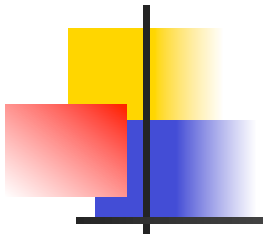
Outline

- File Systems in HPC cluster
- Overview of GPFS features and architecture
- GPFS Cluster Design options
- Quorum
- Installation and Configuration examples
- Exercise
- Management and Overhead Functions

File Systems for HPC clusters

- 3 main categories:
  - Distributed FS (NFS)
    - protocols: TCP/IP
    - Easy to setup and use
    - Single point of failure
    - Do not scale in large clusters
  - SAN or Shared Disk File Systems
    - (protocols: FC, IB, Myricom)
    - Extremely high performance
    - Expensive (each node needs HBA)
  - Parallel FS
    - protocols: any of above
    - Scalable => high performing
    - Redundant => highly available
    - Management requires skilled administrators





Parallel and Distributed FS

- Data distribution:
  - Distributed file systems often store entire objects (files) on a single storage node
  - Parallel file systems distribute data of a single object across multiple storage nodes
- Symmetry:
  - Distributed file systems often use storage is co-located with application
  - In Parallel file systems storage often separated from the compute system
- Fault tolerance:
  - Distributed file systems often use object level data protection (replication) on independent storage devices
  - Parallel FS uses block-level data protection (RAID) on shared storage

Cluster filesystems

Name	By	License	OS	Description
Ceph	Inktank	LGPL	Linux	A massively scalable object store. Ceph was merged into the linux kernel in 2010. Ceph's foundation is the Reliable Autonomic Distributed Object Store (RADOS), which provides object storage via programmatic interface and S3 or Swift REST APIs, block storage to QEMU/KVM/Linux hosts, and POSIX filesystem storage which can be mounted by linux kernel and FUSE clients.
CloudStore	Kosmix	Apache License 2.0		Google File System workalike. Replaced by Quantcast File System (QFS)
Cosmos	Microsoft internal	internal software		Focuses on fault tolerance, high throughput and scalability. Designed for terabyte and petabyte sized data sets and processing with Dryad.
dCache	DESY and others		linux	A write once filesystem, accessible via various protocols
FS-Manager	CDNetworks	proprietary software	Linux	Focused on Content Delivery Network
Gfarm file system	Asia Pacific Grid	X11 License	Linux, Mac OS X, FreeBSD, NetBSD and Solaris	Uses OpenLDAP or PostgreSQL for metadata and FUSE or LUFS for mounting
GPFS	IBM	proprietary	AIX, Linux and Windows	Support replication between attached block storage. Symmetric or asymmetric (configurable)
GlusterFS	Gluster, a company acquired by Red Hat	GNU General Public License v3	Linux, Mac OS X, NetBSD, FreeBSD, OpenSolaris	A general purpose distributed file system for scalable storage. It aggregates various storage bricks over Infiniband RDMA or TCP/IP interconnect into one large parallel network file system. GlusterFS is the main component in Red Hat Storage Server.
Google File System (GFS)	Google	internal software	Linux	Focus on fault tolerance, high throughput and scalability
Hadoop Distributed File System	Apache Software Foundation	Apache License 2.0	Cross-platform	Open source GoogleFS clone

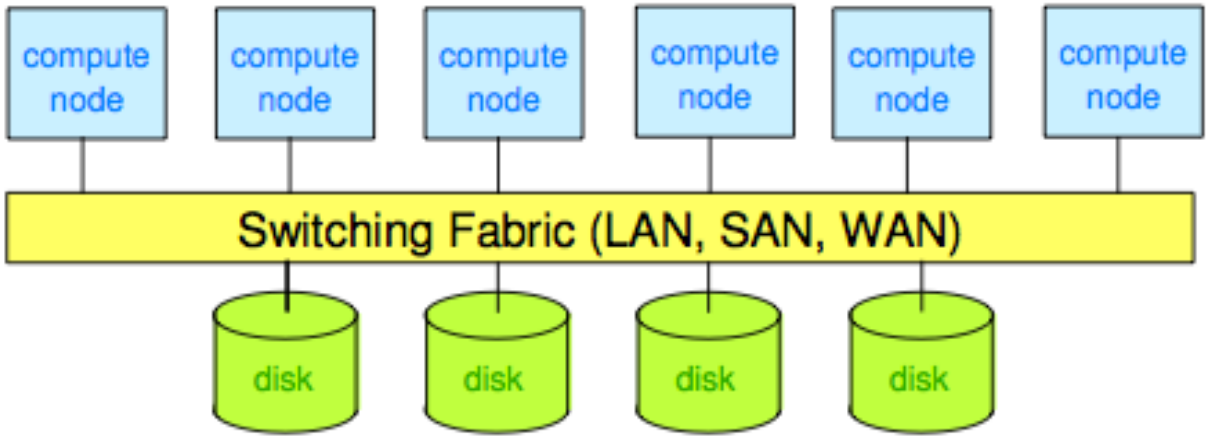
Cluster filesystems

Name	By	License	OS	Description
IBRIX Fusion	IBRIX	proprietary software	Linux	
Lustre	supported by Intel (formerly Whamcloud)	GPL	Linux	A POSIX-compliant, high-performance filesystem. Lustre has high availability via storage failover
MogileFS	Danga Interactive	GPL	Linux (but may be ported)	Is not POSIX compliant, uses a flat namespace, application level, uses MySQL or Postgres for metadata and HTTP for transport.
OneFS distributed file system	Isilon	Proprietary	FreeBSD	BSD based OS on dedicated Intel based hardware, serving NFS v3 and SMB/CIFS to Windows, Mac OS, Linux and other UNIX clients under a proprietary software
Panasas ActiveScale File System (PanFS)	Panasas	proprietary software	Linux	Uses object storage devices
PeerFS	Radiant Data Corporation	proprietary software	Linux	Focus on high availability and high performance and uses peer-to-peer replication with multiple sources and targets
Tahoe-LAFS	Tahoe-LAFS Software Foundation	GNU GPL 2+ and other[19]	Windows, Linux, OS X	secure, decentralized, fault-tolerant, peer-to-peer distributed data store and distributed file system
TerraGrid Cluster File System	Terrascale Technologies Inc	proprietary software	Linux	Implements on demand cache coherency and uses industrial standard iSCSI and a modified version of the XFS file system
XtreemFS [4]	Contrail E.U. project, the German MoSGrid project and the German project "First We Take Berlin"	open-source (BSD)	Linux, Solaris	cross-platform file system for wide area networks. It replicates the data for fault tolerance and caches metadata and data to improve performance over high-latency links. SSL and X.509 certificates support makes XtreemFS usable over public networks. It also supports Striping for usage in a cluster.
MapR-FS	MapR[3]	Proprietary	Linux	Highly scalable, POSIX compliant, fault tolerant, read/write filesystem with a distributed, fault tolerant metadata service. It provides an HDFS and NFS interface to clients

Parallel I/O in a cluster

User **data and metadata** flows between all nodes and all disks **in parallel**

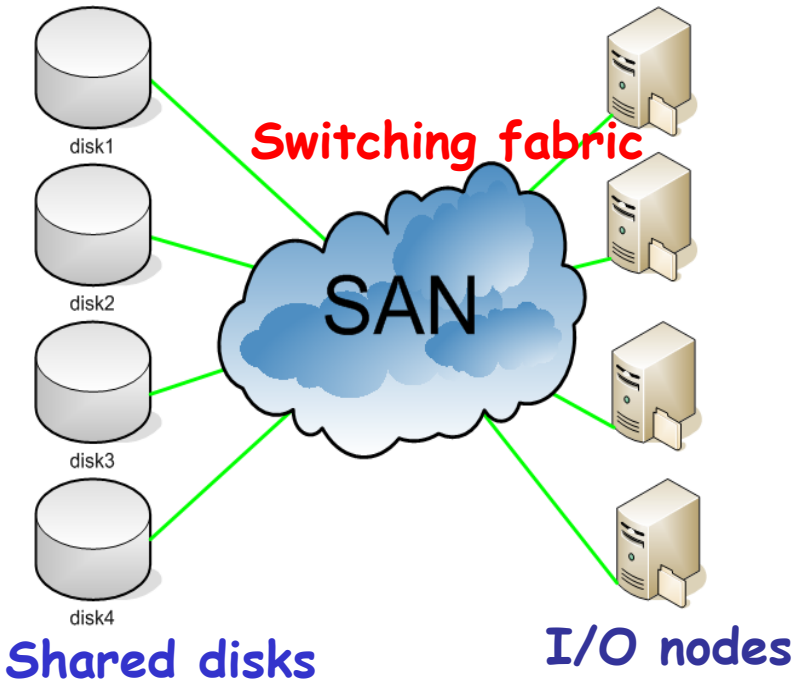
- Multiple tasks distributed over multiple nodes simultaneously access file data
- Multi-task applications access common files in parallel
- Files span multiple disks
- File system overhead operations are distributed and done in parallel
- Provides a consistent global name space across all nodes of the cluster

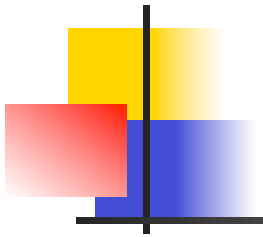


GPFS at glance

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

- *Cluster*: 1..16384 nodes, common admin domain
- *Shared disk*: all data and metadata accessible from any node via disk I/O interface ("any to any" connectivity)
- *Parallel*: data and metadata from all nodes to all disks in parallel
- *Scalable*: 2^{63} files per file system, max file size 2^{99} bytes (= max FS size)
- Real installation:
  - 19 PB in a single file system (T1: 3.5 PB)
  - 400 GB/s of data rate (T1: 16 GB/s)

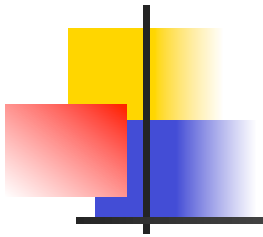




Conceptual understanding

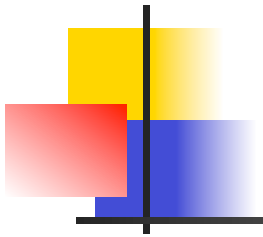
GPFS:

- is not just a Clustered File System software – it's a full featured set of file management tools
- allows a group of computers concurrent access to a common set of data
- the computers can run any mix of AIX, Linux or Windows Server OS
- provides
  - storage management,
  - information life cycle management tools,
  - centralized administration
- allows shared access to file systems from remote GPFS clusters providing a global namespace



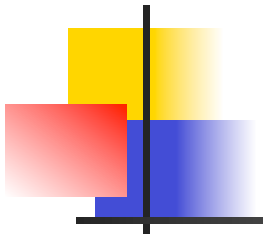
GPFS Cluster

- Trusted domain
- Works on different platforms (OS)
- Provides High Availability (when >1 node)
  - can be configured for continued access to data even if cluster nodes or storage systems fail.
  - built-in tools continuously monitors the health of the file system components
- Supports Shared-disk and from v.3.5 Shared-nothing mode



GPFS File System

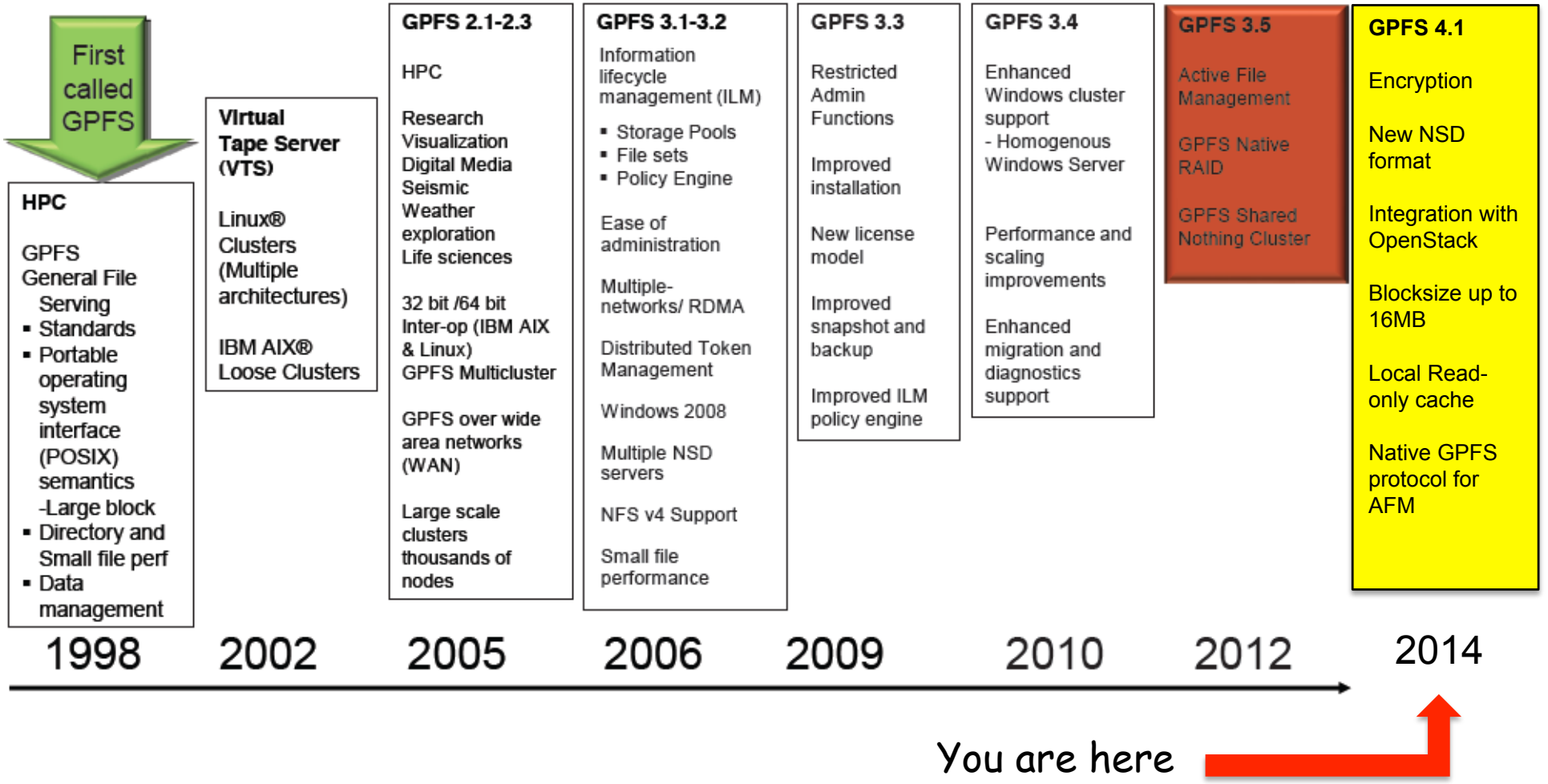
- File Access via standard POSIX file system interfaces
- Supporting wide range of basic configurations and disk technologies
- Journaling
- non-POSIX advanced features
  - e.g., DMAPI, data-shipping, multiple access hints (also used by MPI-IO)
  - ILM, integrated with tape, disaster recovery
  - NFS support (Clustered NFS)
  - Snapshots (copy-on-write)
  - File Placement Optimizer (FPO)
- good performance for large volume, I/O intensive jobs
- Converting to GPFS does not require application code changes (provided the code works in a POSIX compatible environment)

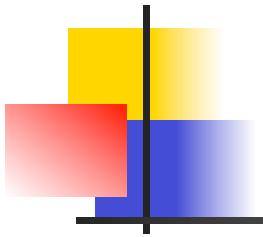


GPFS performance and scalability

- High performance I/O archived by:
  - Striping data across multiple disks attached to multiple nodes.
  - Metadata scans using i-node tree.
  - Chose of file system block sizes (16KB – 16MB) to match applications' I/O requirements .
  - Advanced algorithms for read-ahead and write-behind I/O operations
    - recognizes typical access patterns including sequential, reverse sequential and random
  - Block level locking based on a very sophisticated scalable token management system

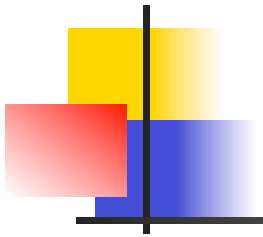
GPFS history and evolution





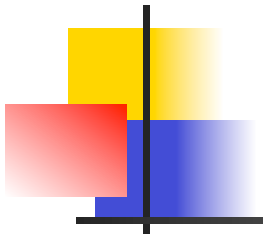
New features in GPFS 3.5

- Active File Management
  - Remote access over unreliable WAN
- High Performance Extended Attributes
  - user-defined attributes stored in i-node
- Independent Filesets
  - Separated i-node spaces
- File cloning
- Shared Nothing Cluster (FPO)
- IPv6 support
- GPFS Native RAID



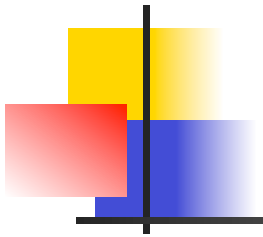
Independent Filesets

- To effectively manage a file system with billions of files requires advanced file management technologies.
- independent fileset has its own i-node space.
  - independent fileset can be managed similar to a separate file system but still allow you to realize the benefits of storage consolidation.
- only needs to scan the i-node space represented by that fileset,
  - so if you have 1 billion files in your file system and a fileset has an i-node space of 1 million files, the scan only has to look at 1 million i-nodes. This instantly makes the policy scan much more efficient.
- Independent filesets enable other new fileset features in GPFS 3.5.
  - **Fileset Level Snapshots**
    - Snapshot granularity is now at the fileset level in addition to file system level snapshots.
  - **Fileset Level Quotas**
    - User and group quotas can be set per fileset



Other features

- File Cloning
  - File clones are space efficient copies of a file where two instances of a file share data they have in common and only changed blocks require additional storage. File cloning is an efficient way to create a copy of a file, without the overhead of copying all of the data blocks.
- IPv6 Support
  - IPv6 support in GPFS means that nodes can be defined using multiple addresses, both IPv4 and IPv6.
- Independent metadata block size
  - Up to 1/32 of data block size

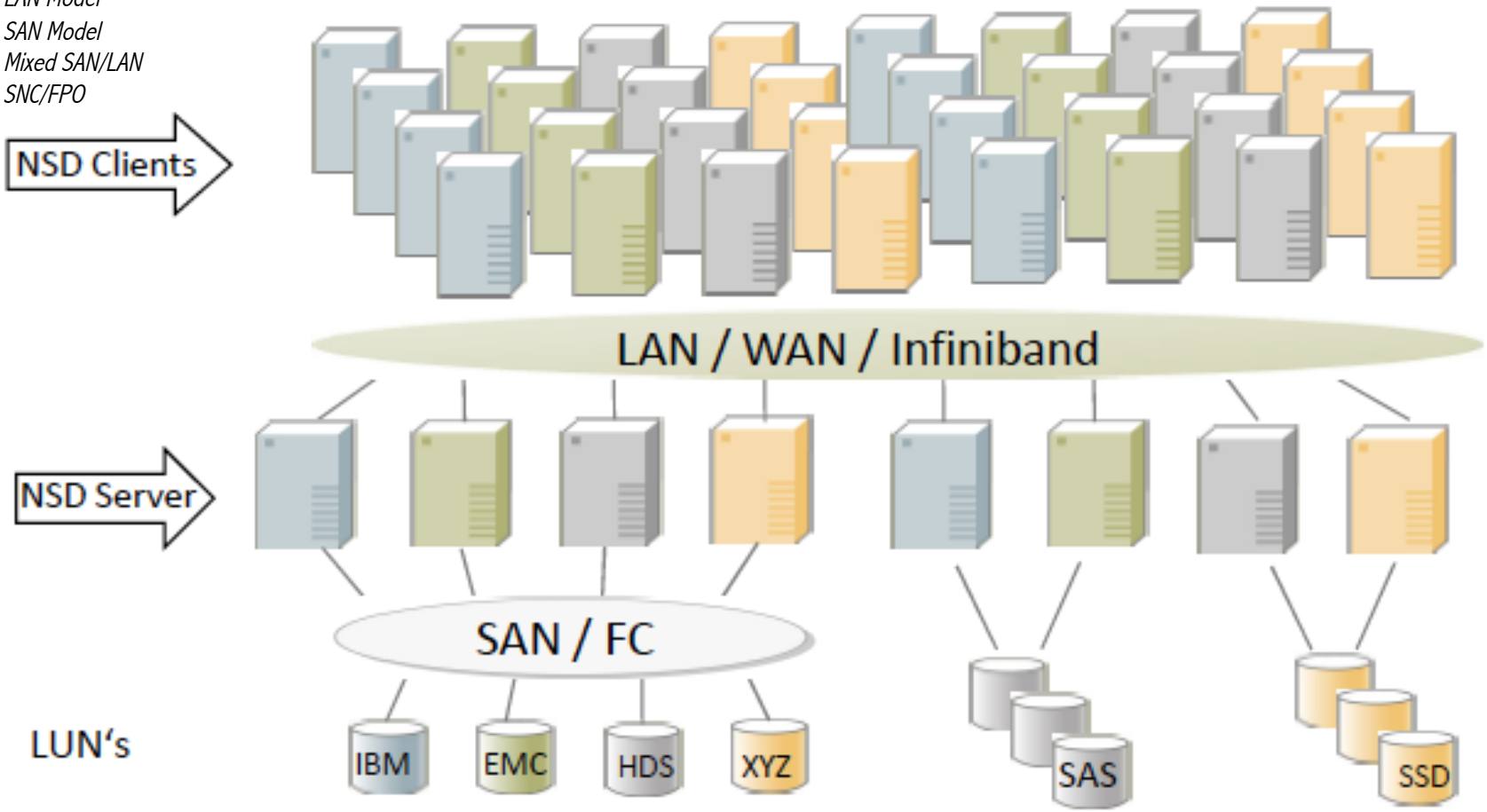


GPFS Native RAID

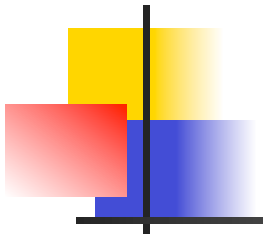
- GPFS brings storage RAID management into the GPFS NSD server.
- With GNR GPFS directly manages JBOD based storage.
  - This feature provides greater availability, flexibility and performance for a variety of application workloads.
- GNR implements a Reed-Solomon based de-clustered RAID technology
  - provide high availability and keep drive failures from impacting performance by spreading the recovery tasks over all of the disks.
  - Unlike standard Network Shared Disk (NSD) data access GNR is tightly integrated with the storage hardware.
  - For GPFS 3.5 GNR is available only on the IBM HW platforms

GPFS Architecture

LAN Model
SAN Model
Mixed SAN/LAN
SNC/FPO



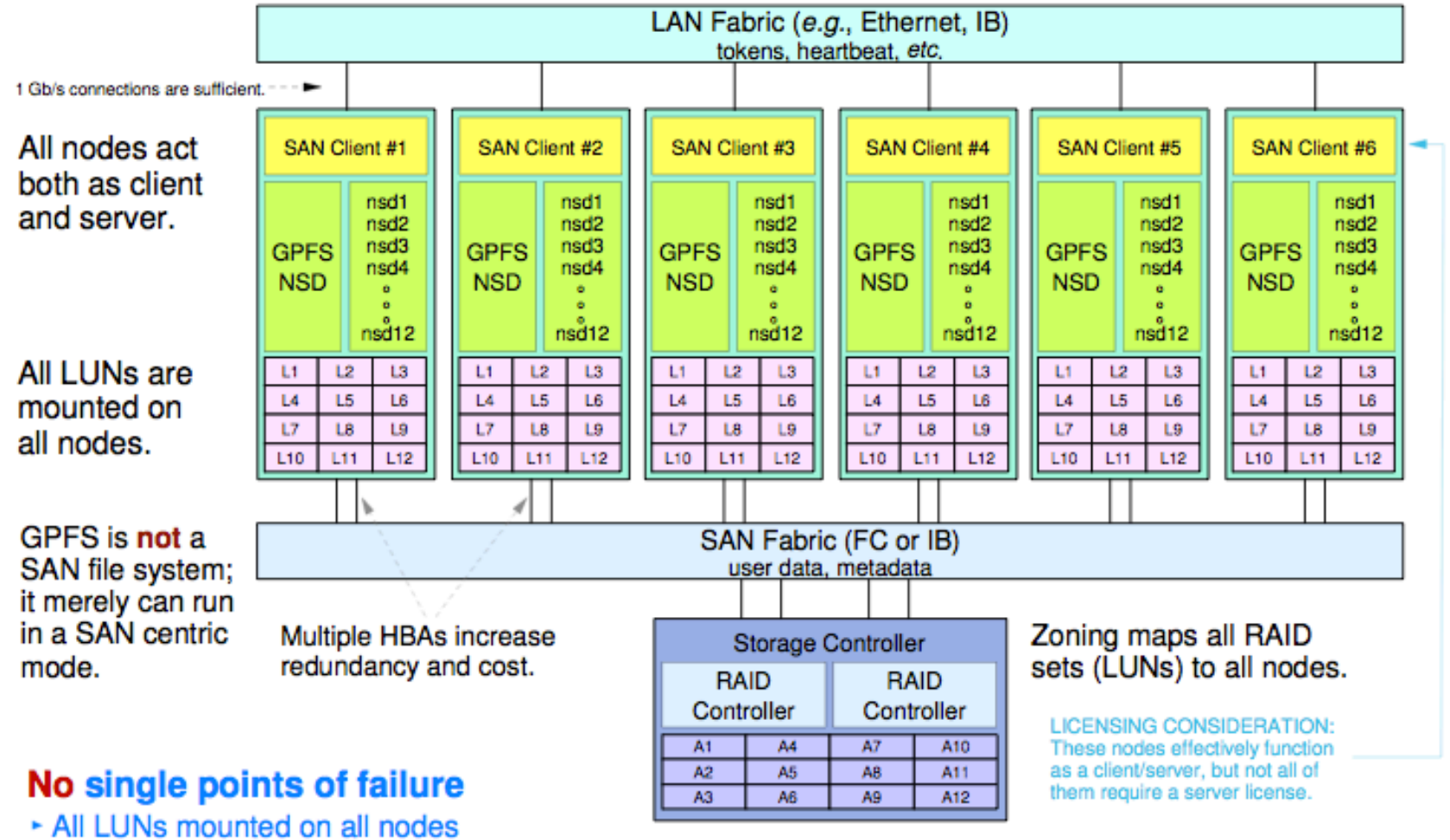
(NSD = Network Shared Disk)



GPFS cluster components

- **Node** – individual OS instance within cluster
- **Network Shared Disk** (**NSD**) – a storage device (LUN) available to the cluster to be used to build a file system
- **NSD server** – I/O server which provides access to a particular **NSD**
- **GPFS** file system – build from set of **NSD**s
- **Application** node – node which mounts GPFS file system and run applications accessing the file system

SAN topology



No single points of failure

- All LUNs mounted on all nodes
- SAN connection (FC or IB) fail over
- Dual RAID controllers

The largest SAN topologies in producton today are 256 nodes, but require special tuning.

CAUTION:

A SAN configuration is ***not*** recommended for larger clusters (*e.g.*, >= 64 since queue depth must be set small (*e.g.*, 1)

LAN topology

NSD

- SW layer in GPFS providing a "virtual" view of a disk
- virtual disks which correspond to LUNs in the NSD servers with a bijective mapping

LUN

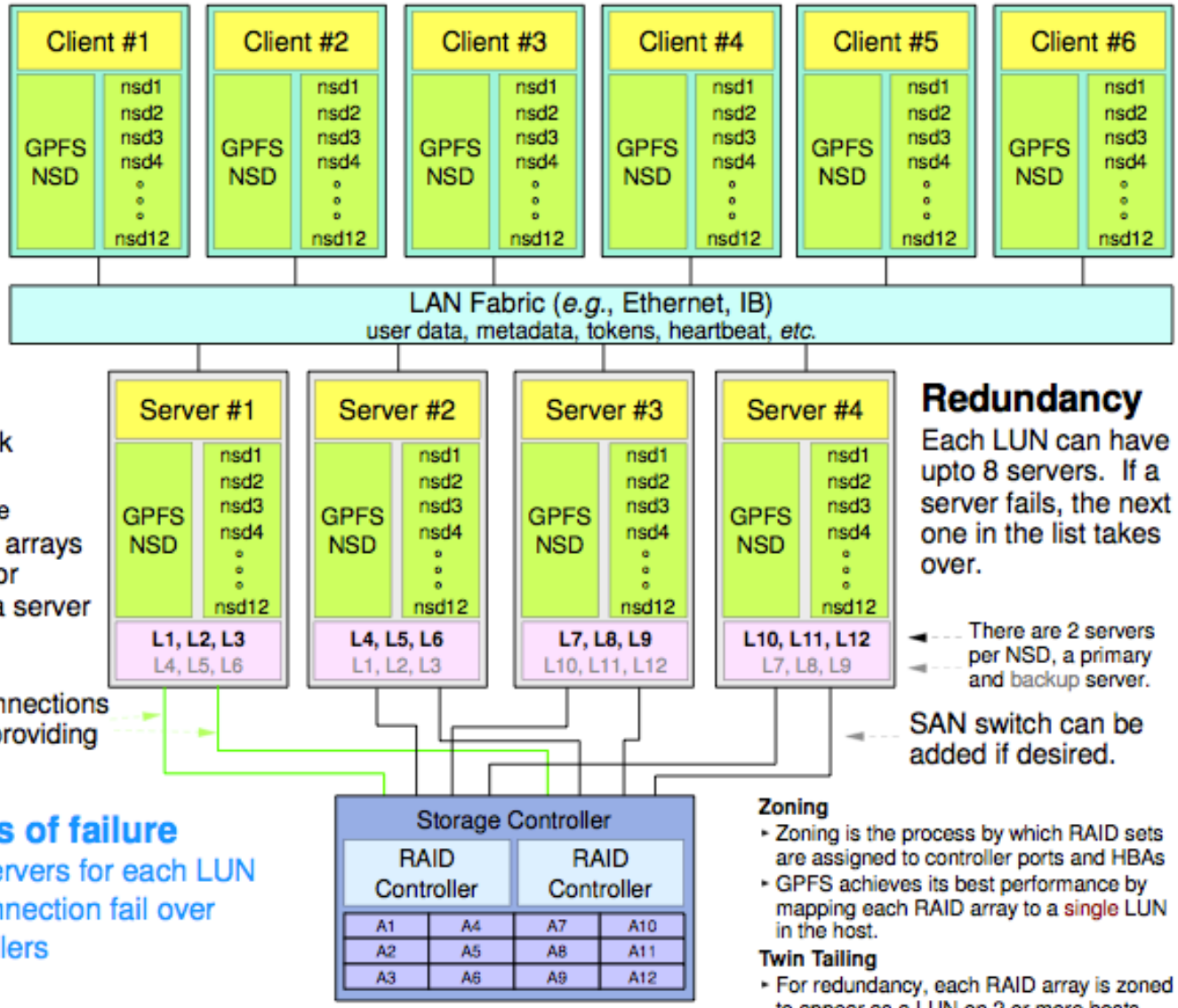
- Logical Unit
- Abstraction of a disk
  - AIX - hdisk
  - Linux - SCSI device
- LUNs map to RAID arrays in a disk controller or "physical disks" in a server

Redundancy

Each server has 2 connections to the disk controller providing redundancy

No single points of failure

- primary/backup servers for each LUN
- controller/host connection fail over
- Dual RAID controllers



Redundancy

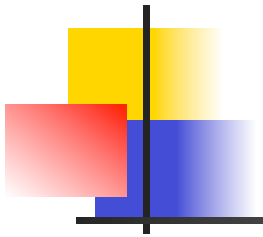
Each LUN can have upto 8 servers. If a server fails, the next one in the list takes over.

Zoning

- Zoning is the process by which RAID sets are assigned to controller ports and HBAs
- GPFS achieves its best performance by mapping each RAID array to a single LUN in the host.

Twin Tailing

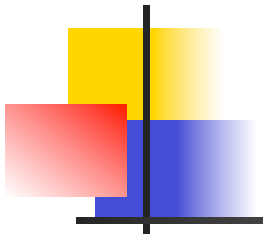
- For redundancy, each RAID array is zoned to appear as a LUN on 2 or more hosts.



Comparing LAN and SAN Topologies

LAN Topology

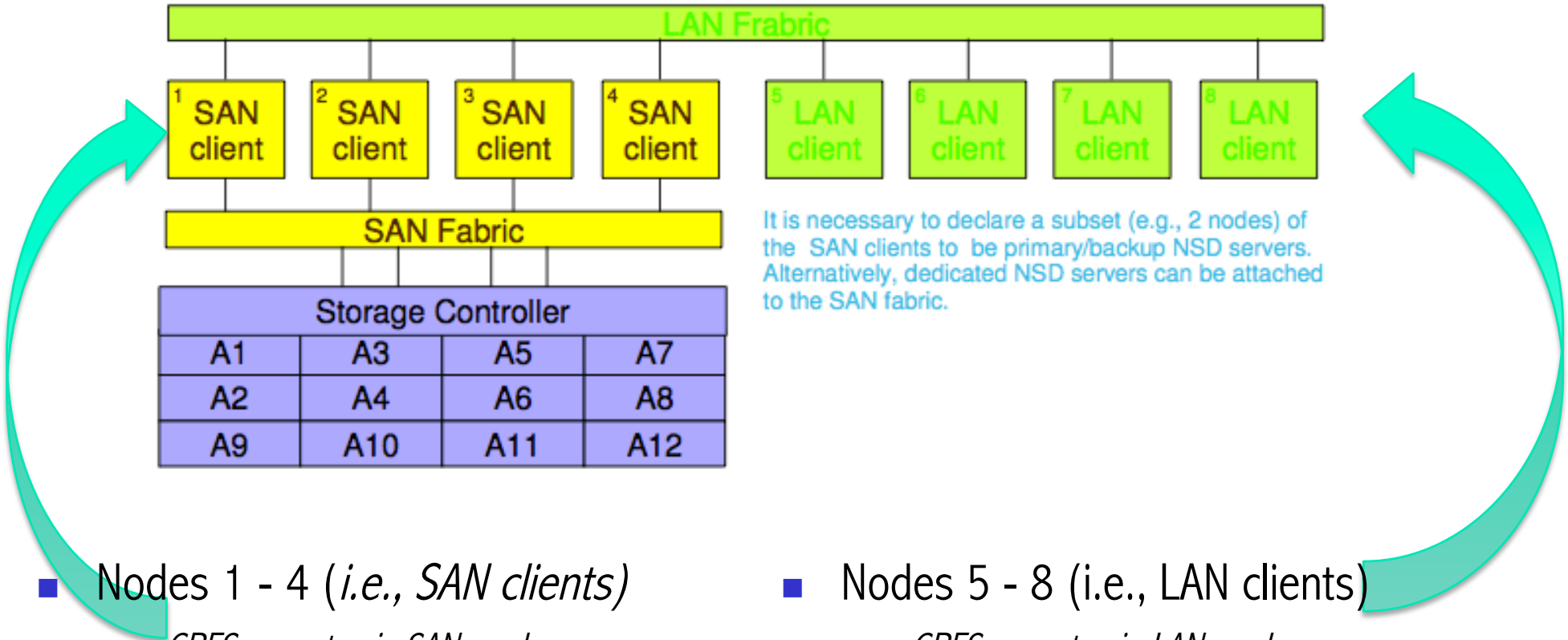
- **All GPFS traffic** (user data, metadata, overhead) traverses **LAN** fabric
- **Disks** attached only to **servers** (NSD servers)
- **Applications** generally run only on the **clients** (GPFS clients); however, applications can also run on servers
  - cycle stealing on the server can adversely affect synchronous applications
- Economically scales out to large clusters
  - ideal for an "army of ants" configuration (*i.e., large number of small systems)*
- *Potential bottleneck: LAN adapters*
  - *e.g., GbE adapter limits peak BW per node to 120 MB/s; "channel aggregation" improves BW*



Comparing LAN and SAN Topologies (3)

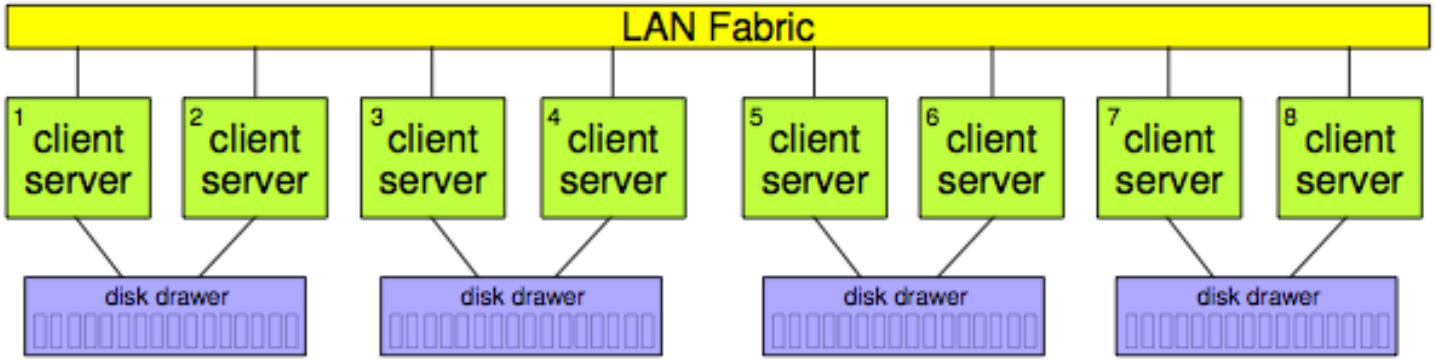
- From application point of view data access on Network attached nodes is exactly the same as on SAN attached node.
- GPFS transparently sends the block-level I/O over TCP/IP network.

Mixed LAN/SAN topology



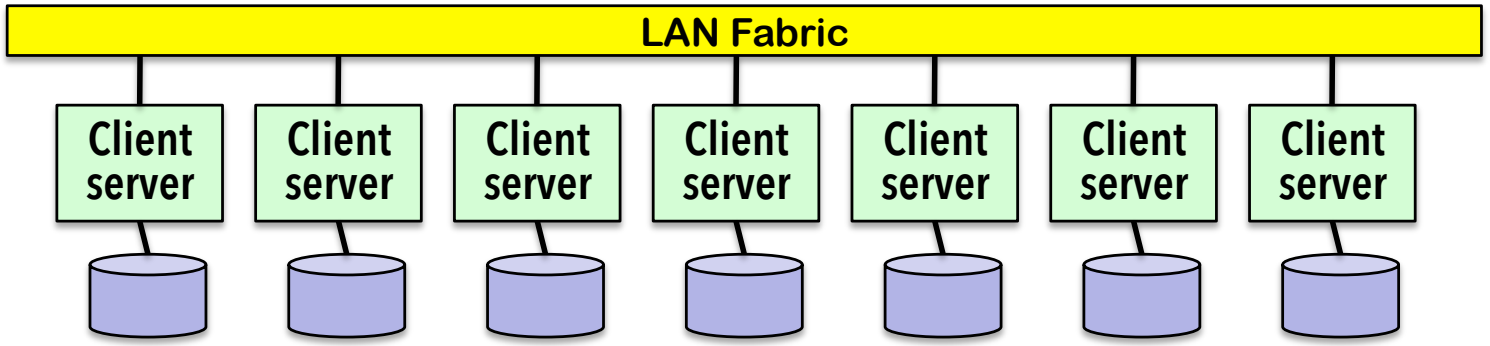
- Nodes 1 - 4 (*i.e., SAN clients)*
  - *GPFS operates in SAN mode*
  - *User and meta data traverse the SAN*
  - *Tokens and heartbeat traverse the LAN*
- Nodes 5 - 8 (i.e., LAN clients)
  - *GPFS operates in LAN mode*
  - *User data, meta data, tokens, heartbeat traverse the LAN*

Symmetric clusters

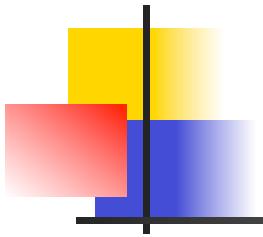


- No distinction between NSD clients and NSD servers
- Provides excellent scaling and performance
- For use with "twin tailed disk" (SAS) to avoid single point of failure risks or attach servers to storage controllers without fabric switch

Shared Nothing clusters

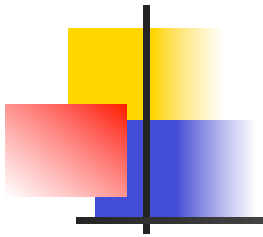


- Since v.3.5
- No distinction between NSD clients and NSD servers
- Hadoop paradigm
- Increased level of replication: up to 3 replicas in FPO pool
  - Auto recovery from a node or disk failure
- 3-D Failure group definition (x,y,z)



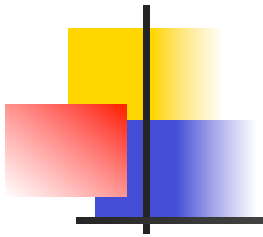
Performance related features in GPFS

- Multithreading
- Striping
- File caching
- Byte range locking
- Blocks and sub-blocks
- Access pattern optimization



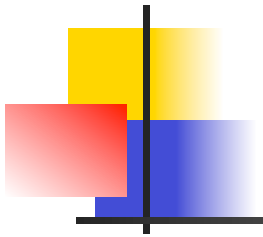
Multithreaded Architecture

- GPFS can spawn up to
  - 512 threads/node for 32 bit kernels
  - 1024 threads/node for 64 bit kernels
    - there is one thread per block (i.e., each block is an IOP)large records may require multiple threads
- The key to GPFS performance is "deep prefetch" which due its multithreaded architecture and is facilitated by
  - GPFS pagepool
  - striping (which allows multiple disks to spin simultaneously)
  - access pattern optimizations for sequential and strided access or the explicit use of hints



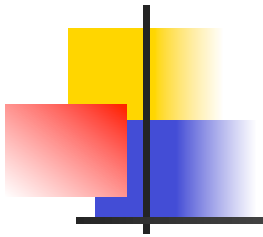
Data Striping

- GPFS stripes successive blocks of each file across successive disks
- Disk I/O for sequential reads and writes is done in parallel (prefetch, write behind)
- Make no assumptions about the striping pattern
- Block size is configured when file system is configured, and is not programmable
- transparent to programmer



Quorum

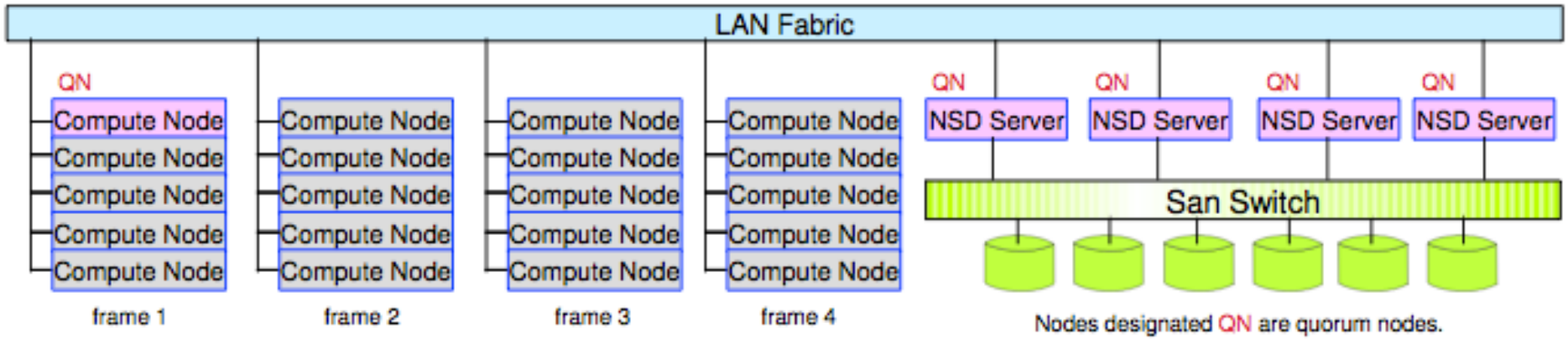
- Problem
  - If a key resource fails (*e.g., cluster manager or token manager) GPFS will spawn a new one to take over. But if the other one is not truly dead (e.g., network failure), this could create 2 independent resources and corrupt the file system.*
- *Solution*
  - *Quorum must be maintained to recover failing nodes.*
  - *2 options*
    - *Node quorum (default): must have at least 3 quorum nodes*
    - *Node quorum with tiebreaker disks: used in 1 or 2 node clusters*



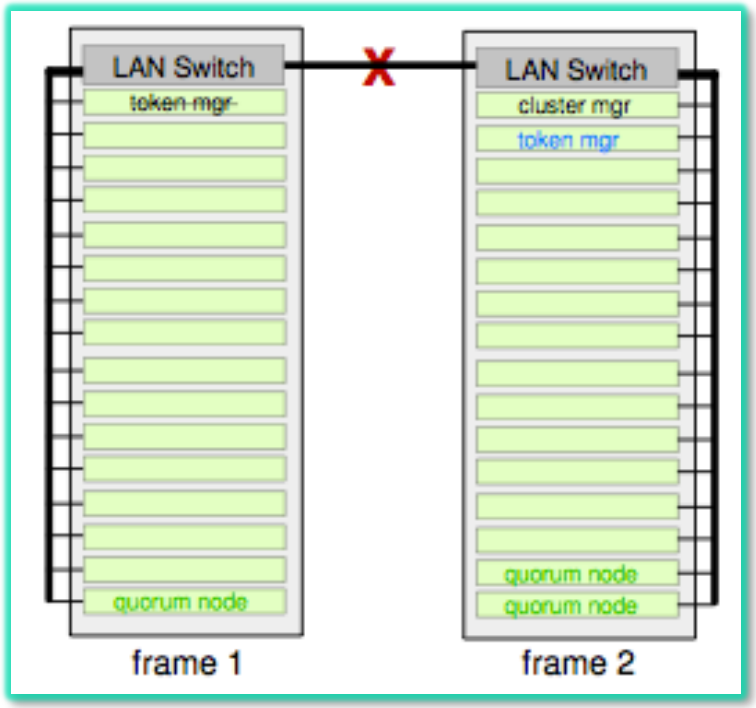
Quorum

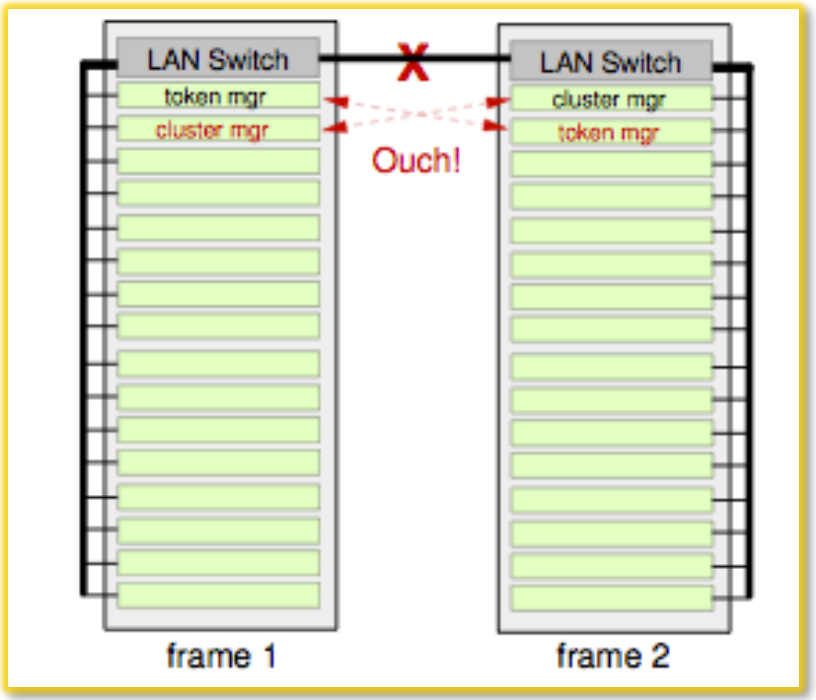
- Node quorum is defined as one plus half of the explicitly defined quorum nodes in the GPFS cluster. There are no default quorum nodes. The smallest node quorum is 3 nodes.
- Selecting quorum nodes: best practices
  - Select nodes most adapted to remain active
  - Select nodes that rely on different failure points
    - example: select nodes in different racks or on different power panels.
  - In smaller clusters (*e.g., < 1000 nodes) select administrative nodes*
    - *common examples: NSD servers, login nodes*
  - *In large clusters, either select dedicated nodes or overlap with manager nodes.*
    - *do not overlap with NSD servers*
  - *Select an odd number of nodes (e.g., 3, 5, or 7 nodes)*
    - *More than 7 nodes is not necessary; it increases failiure recovery time without increasing availability.*

Quorum


LAN Fabric
QN
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
frame 1
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
frame 2
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
frame 3
Compute Node
Compute Node
Compute Node
Compute Node
Compute Node
frame 4
QN
NSD Server
QN
NSD Server
QN
NSD Server
QN
NSD Server
San Switch
Nodes designated QN are quorum nodes.

Split brain

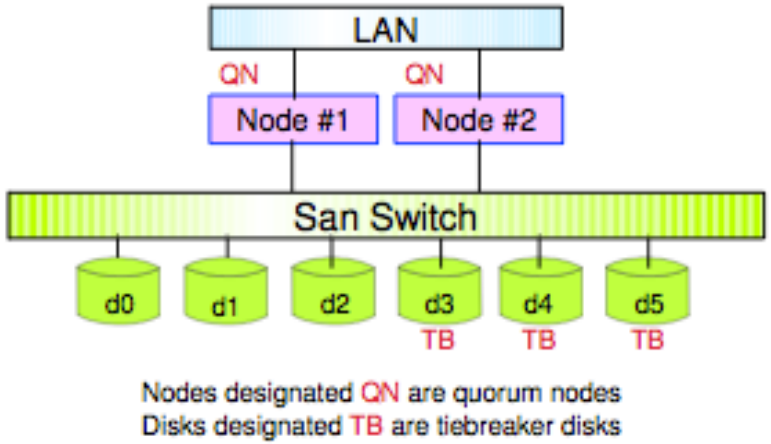




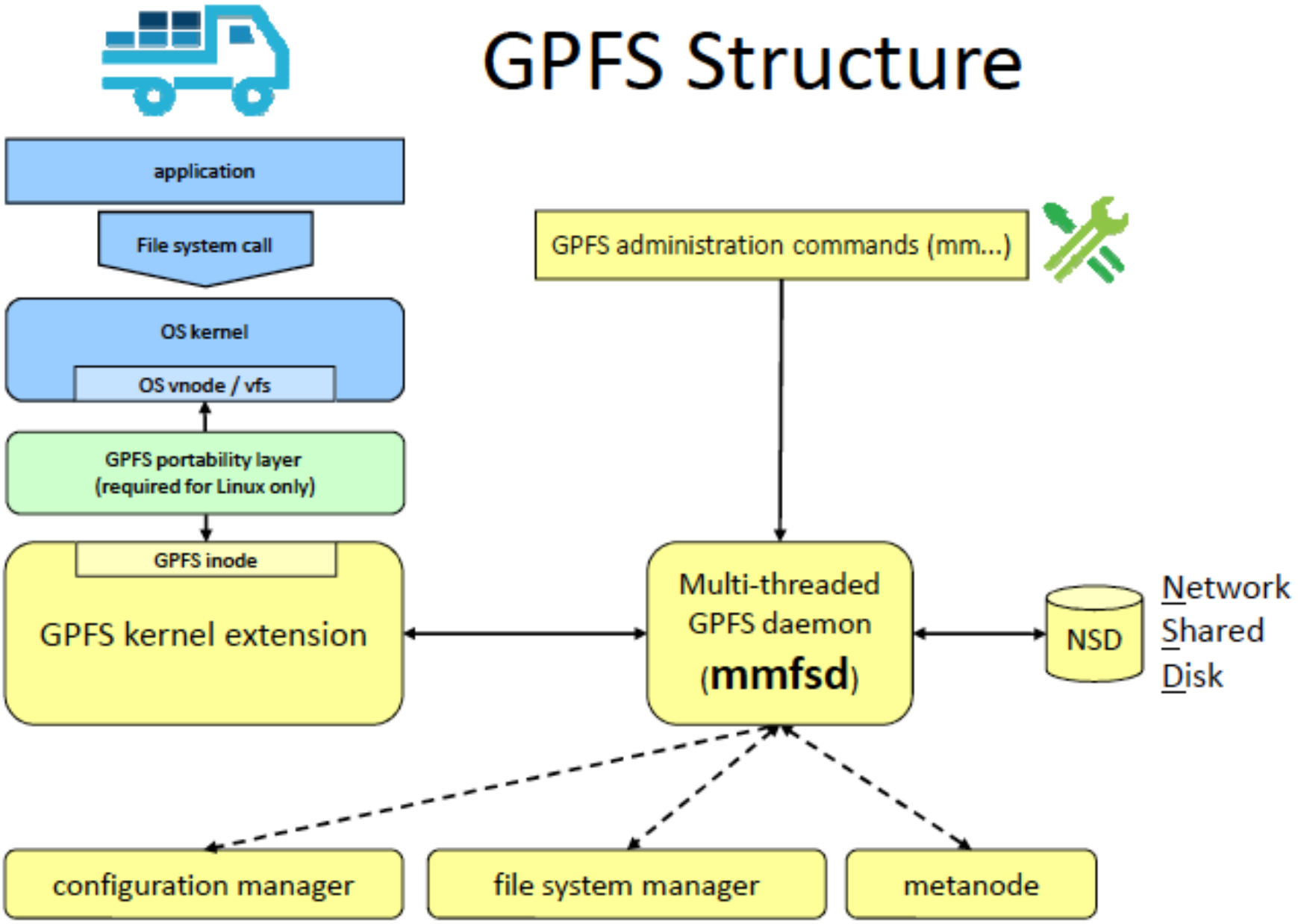
- Without the quorum rule:
  - frame 1 could start an independent cluster manager
  - frame 2 could start an independent token manager
- file system would be corrupted!

Node Quorum with Tiebreaker Disks

- Maximum 8 quorum nodes
- runs with as little as **one quorum node** (minQuorumNodes) available so long as there is access to a *majority of the quorum disks.*
- the number of non-quorum nodes is unlimited (can be as small as zero)
- there can be 1 to 3 tiebreaker disks (n.b., odd number of disks is best)
  - tiebreakers disks must be directly accessible from the quorum nodes, but do not have to belong to any particular file system
  - must have a cluster-wide NSD name as defined through the mmcrnsd command
  - tiebreaker disks must be SAN attached (FC or IP) or VSDs
- same rules apply in selecting quorum nodes for both quorum options
- select quorum nodes with mmcrcluster or mmchconfig commands
- select tiebreaker disks with mmchconfig command (cluster down)



GPFS Structure



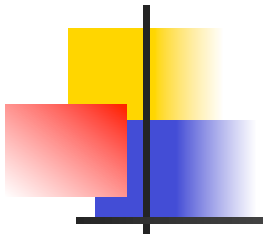
Where GPFS lives?

- 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
- Almost all GPFS commands starts with "mm"

Add it to your PATH

The first place to look in case of problems

Main cluster configuration (must be the same on all nodes)



Example: creating GPFS Cluster

- **mmcrcluster** - Creates a GPFS cluster from a set of nodes.

```
>mmcrcluster -N gpfs.nodelist \          Primary config Server
             -p c1-serv1 \               Secondary config server
             -s c1-serv2 \               Remote shell command
             -r /usr/bin/ssh \           Remote copy command
             -R /usr/bin/scp \           Cluster name
             -C c1.openlab.infn.it \     Domain name
             -U openlab.infn.it
>
>cat gpfs.nodelist
C1-serv1:quorum-manager
C2-serv2:quorum-manager
C3-serv3:quorum
```

Startup and shutdown

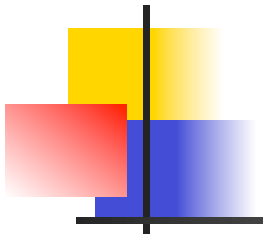
- Before you start cluster define license for each node:

```
# mmchlicense server –accept –N gpfs.nodelist
```

- **mmstartup** and **mmshutdown**
  - startup and shutdown the **mmfsd** daemons
  - if necessary, mount file system after running **mmstartup**
    - properly configured, mmfsd will startup automatically (n.b., no need to run **mmstartup**); if it can not start for some reason, you will see **runmmfs** running and a lot of messages in /var/adm/ras/mmfs.log.latest
  - mmgetstate - displays the state of the GPFS daemon on one or more nodes:

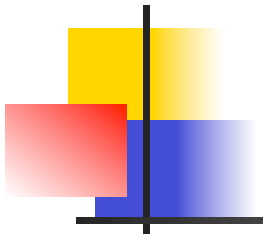
```
# mmgetstate -a

 Node number  Node name        GPFS state
------------------------------------------
       1      gpfs-01-01       active
       2      gpfs-01-02       active
       3      gpfs-01-03       active
       4      TSM-TEST-1       active
```



Exercise 1: Create GPFS Cluster and a Filesystem

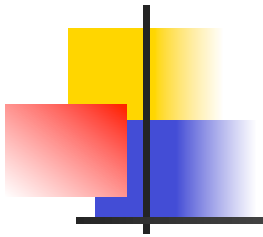
- Commands to use:
  - # mmcrcluster
  - # mmlscluster
  - # mmchlicense
  - # mmstartup
  - # mmgetstate
  - # mmcrnsd
  - # mmcrfs
  - # mmmount
  - # mmlsmount
- Check man pages if in doubt



GPFS File Caching

GPFS Pagepool

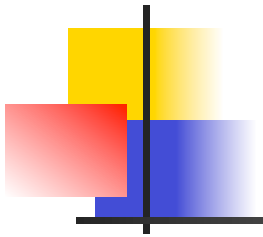
- What is the pagepool?
  - It is a pinned memory cache used exclusively by GPFS
    - The pagepool is independent of the VMM subsystem
      - vmtune has no direct impact on the pagepool
    - GPFS uses mmap, schmat or kernel calls to do the pinning operation
  - It is used by GPFS for file data, indirect blocks and "system metadata" blocks
    - map blocks i-nodes in transit
    - recovery log buffers
    - emergency buffers



GPFS pagepool

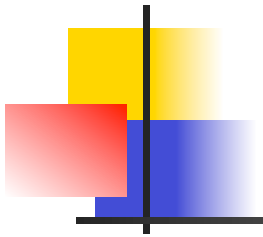
- Pagepool size Set by
  `mmchconfig pagepool= {value}`
  - default: 1GB
  - min value: 16M These values are generally too small, especially for large blocks (*e.g., 4M)*
  - max value:
    - 256G for 64 bit OS,
    - 2G for 32 bit OS
- BUT GPFS will not
  - allocate more than the pagepool parameter setting
  - allocate more than 75% of physical memory
    - This can be changed to values between 10% to 90% *e.g.,*

    `mmchconfig pagepoolMaxPhysMemPct=90`
  - request more memory than the OS will allow



Optimum size of the pagepool

- Best determined empirically
  - Optimum pagepool sizing is partially workload dependent
  - File systems with a large blocksize and/or a larger number of LUNs requires a larger pagepool
    - assume blocksize <= 1 MB and number of LUNs <= 12, then let sizeof(pagepool) <= 256 MB
    - assume blocksize >= 2 MB and number of LUNs >= 24, then let sizeof(pagepool) >= 512 MB
  - Optimizing streaming access requires a smaller pagepool (*e.g., up to 1 GB)*
  - *Optimizing irregular access requires a larger pagepool (e.g., > 1 GB, enough to hold working set)*

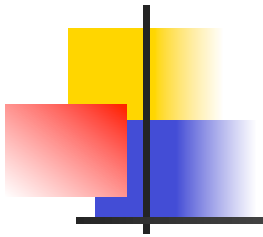


GPFS pagepool

Miscellaneous observations

- Large pagepools are most helpful when
  - writes can overlap computation
  - heavy reuse of file records (*n.b., good temporal locality)*
  - *semi-random access patterns with acceptable temporal locality*
  - *a GPFS node is used as a login node or an NFS server for large clusters*
- *Pagepool size on NSD servers*
  - *general principle: NSD servers do not cache data; they use the pagepool for transient buffers*
  - *Formula*

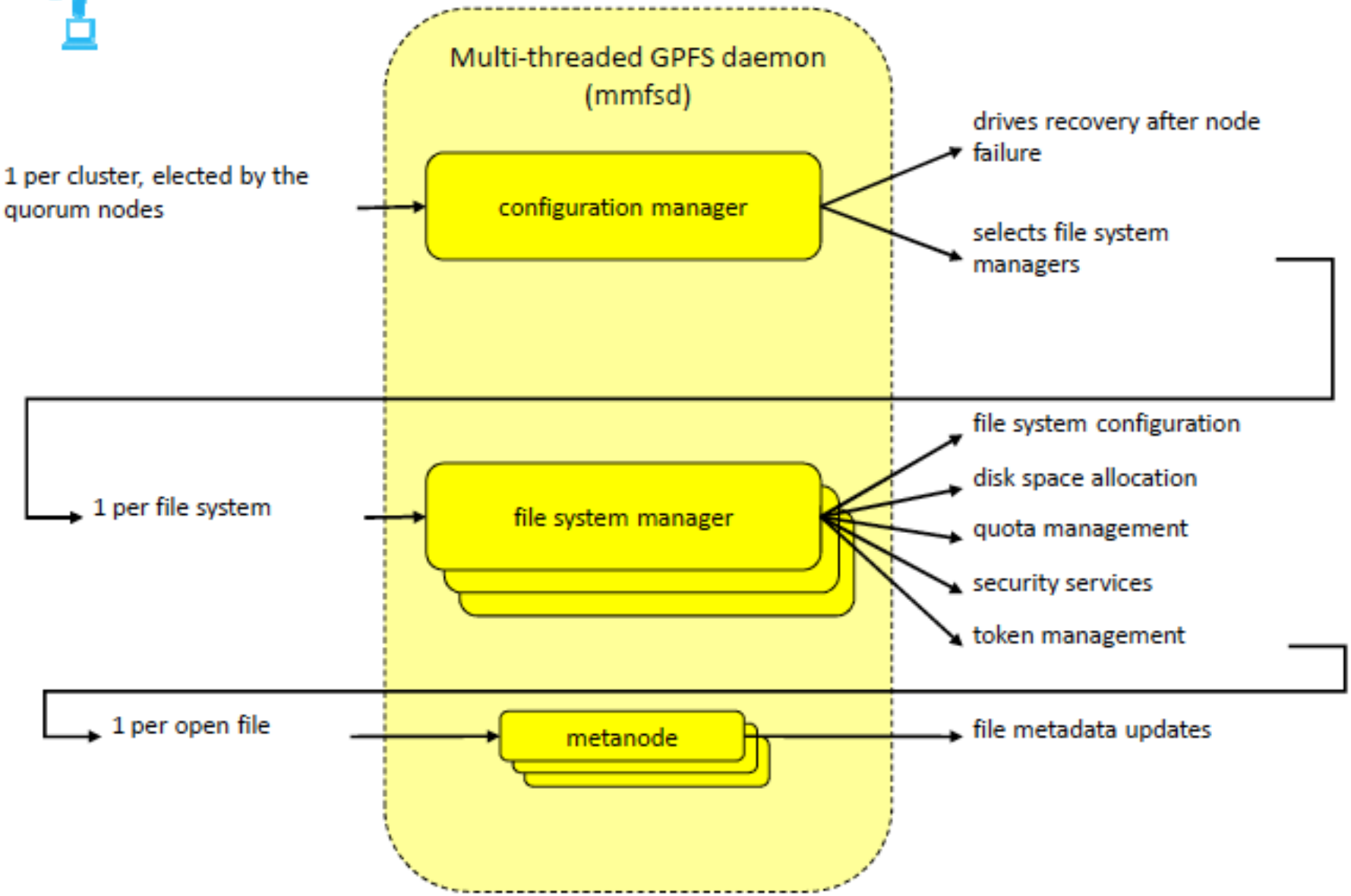
pagepool_size = largest blocksize * NSDThreads NSDThreads = min(A1, max(A2, A3)) A1 = nsdMaxWorkerThreads A2 = nsdMinWorkerThreads A3 = K * nsdThreadsPerDisk K = number of LUNs per NSD server

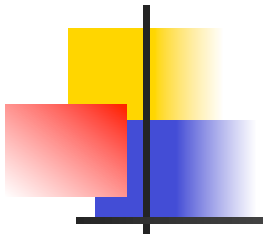


GPFS Management and Overhead Functions

- Metanode
- Configuration Manager
  - AKA "cluster configuration manager"
- Cluster Manager
- Manager Nodes
  - File System Managers
    - File system configuration
    - Manage disk space allocation
    - Quota management
    - Security services Token Managers
- Quorum Nodes

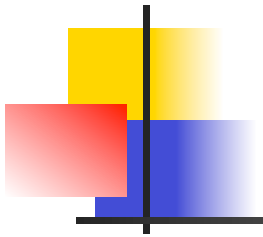
GPFS daemon roles





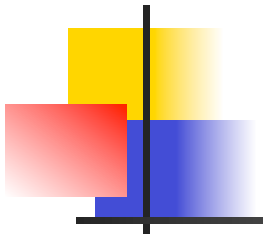
Metanode

- (one per file) collects file size, mtime/atime, and indirect block updates from other nodes
- is elected dynamically and can move dynamically
- Only the metanode reads & writes i-node and indirect blocks
- Merges i-node updates by keeping largest file size and latest mtime
- Does Synchronization
  - Shared write lock allows concurrent updates to file size and mtime.
  - Operations that require exact file size/mtime (e.g., stat) conflict with the shared write locks.
  - Operations that may decrease file size or mtime



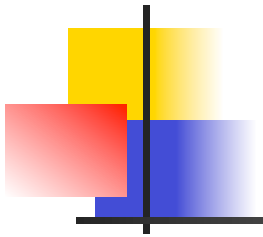
Configuration Manager

- There is a primary and backup configuration manager per GPF Scluster
  - Specified when the cluster is created using mmcrcluster
  - Common practice
    - assign to manager nodes and/or NSD servers.
  - Function
    - Maintains the GPFS configuration file /var/mmfs/gen/mmsdrfs on all nodes in the GPFS cluster. This configuration file can not be updated unless both the primary and backup configuration managers are functioning.
  - Minimal overhead



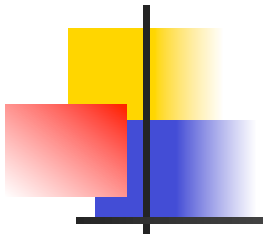
Cluster Manager

- There is one cluster manager per GPFS cluster
- Selected by election from the set of quorum nodes
  - can be changed using mmchmgr
- Functions
  - Monitors disk leases (*i.e., "heartbeat")*
  - *Detects failures and directs recovery within a GPFS cluster*
  - *determines whether quorum exists*
  - *Manages communications with remote clusters*
    - *distributes certain configuration changes to remote clusters*
    - *handles GID/UID mapping requests from remote clusters*
  - *Selects the file system manager node*
    - *by default, it is chosen from the set of designated manager nodes*
    - *choice can be overridden using mmchmgr or mmchconfig commands*



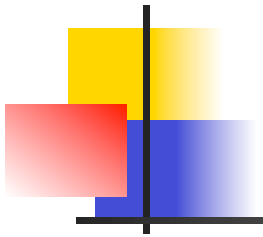
Cluster manager (2)
network considerations

- Heartbeat network traffic is light and packets are small
  - default heartbeat rate = 1 disk lease / 30 sec per node
  - cluster manager for a 4000 node cluster receives 133 disk leases per second
- But network congestion must not be allowed to interfere with the heartbeat
  - by default, disk lease lasts 35 sec, but a node has last 5 sec to renew lease
  - best practice: assign to a lightly used or dedicated node in clusters over 1000 nodes



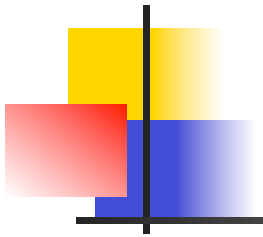
Manager Nodes

- Designating manager nodes
  - They are specified when a cluster is created (using mmcrcluster)
    - can be changed using mmchnode
  - Can specify up to 128 manager nodes
    - If not specified, GPFS selects 1 node to be a combined file system and token manager node.
- Function
  - File system managers
  - Token managers
- Best practices
  - smaller clusters (less than 1000 nodes):
    - commonly overlaped with NSD servers and/or quorum nodes
  - larger clusters (more than 1000 nodes):
    - assign to lightly used or dedicated nodes do not overlap with NSD servers



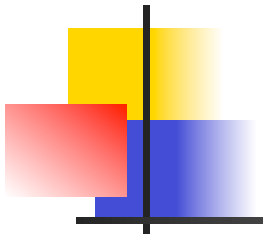
File System Managers

- There is exactly one file system manager per file system
  - File system managers are uniformly distributed over manager nodes
    - A file system manager never spans more than 1 node, but if there are more file systems than manager nodes, there will be multiple file system managers per manager node.
  - Choice can be overridden by mmchmgr command
    - any node can be chosen (*n.b., it does not have to be a manager node)*
- File system manager functions (see next slide)
- Low overhead



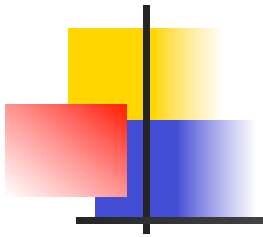
File system manager functions

- file system configuration
  - adding disks
  - changing disk availability
  - repairing the file system
  - mount/umount processing (this is also done the node requesting the operation)
- disk space allocation management
  - controls which regions of each disk are allocated to each node (striping management)
- quota management
  - enforces quotas if it has been enabled (see mmcrfs and mmchfs commands)
  - allocates disk blocks to nodes writing to the file system
  - generally more disk blocks are allocated than requested to reduce need for frequent requests



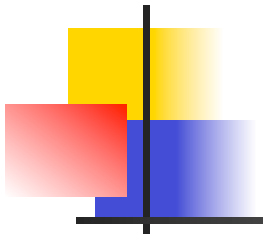
Token Managers

- Token managers run on manager nodes
  - GPFS selects some number of manager nodes to run token managers
    - GPFS will only use manager nodes for token managers
    - the number of manager nodes selected is based on the number of GPFS client nodes
  - Token state for each file system is uniformly distributed over the selected manager nodes
    - there is 1 token manager per mounted file system on each selected manager node
  - 1 manager node can process >= 500,000 "tokens" using default settings
    - In this context, a token is a set of several tokens ~= 600 bytes on average.
    - total number of tokens = number of nodes * (maxFilesToCache + maxStatCache) + all currently open files
    - If the selected manager nodes can not hold all of the tokens, GPFS will revoke unused tokens, but if that does not work, the token manager will generate an ENOMEM error. (This usually happens when not enough manager nodes were desiginated.)



Token Managers

- Overhead
  - CPU usage is light
  - Memory usage is light to moderate
    - (*e.g., at most 512 MB by default) can be changed using* **`mmchconfig tokenMemLimit=<value>`**
  - *Message traffic is variable, but not excessive. It is characterized by many small packets*
    - *If network congestion impedes token traffic, performance will be compromised, but it will not cause instability.*
    - *If NSD servers and GPFS clients are also used for token management, large block transfers (e.g., >= 512 KB)* may impede token messages.
      - If these issues are impeding token response, chances are good that users will never notice.



Servers and clients

- *server functions are commonly overlapped ← This reduces cost, but use caution!*
  - *example: use NSD servers as quorum and manager nodes*
- *client licenses cost less than server licenses. ← The new licensing model is much cheaper!*
- *server nodes can perform client actions, but client nodes can not perform server actions*