IBMCHAMPION

Belisama

Session: s111035 Introduction to GPFS

Antony Steel

Thanks to:
Sven, Scott and the GPFS team
Lin Feng Shen

Power Systems

Abstract

- This session will introduce participants to the installation, configuration, operation and management of Spectrum Scale (GPFS)
- Explore some of new features of Spectrum Scale, ESS, ECE ...

Spectrum Scale (GPFS)

Belisama

- Introduction
  - File System Types
  - History and Milestones
  - Usage scenarios
  - Key strengths and competition
- Understanding GPFS
  - Base Concepts
  - Network Shared Disks (NSD)
  - Blueprints & Daemons
- Get Started / How To Guide for AIX/Linux
  - Creating a GPFS Cluster
  - Network Shared Disk infrastructure
- GPFS features and functions

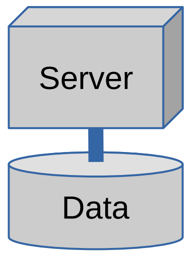
Spectrum Scale / GPFS, what is it?

- What is it
  - Software from IBM that runs on AIX, Linux (p/x/z), Windows (Client / Server)
  - Serves data (file, object..) via GPFS “protocol” (Client Software), NFS, SMB, Swift
  - IBM also sells as
    - Appliance: ESS (older versions SoFS and SoNAS)
    - Solutions: DB2 PureScale, HPC, AI, SAP, NovaLink, Oracle RAC
- What does it include
  - High performance scalable posix file system
  - Management GUI (for management and monitoring)
  - Powerful command line and API
  - Integration with other tiers of storage (tape / cloud)
- IBM’s best kept secret
  - Originally designed for multimedia applications on SP, disappeared from view as HPC solution
  - Reappeared in commercial space to handle:
    - Explosion in the growth of unstructured data
    - Old, expired, unused data occupying space on expensive storage
    - Single file stores filling up, not meeting the increasing demand for throughput or management ease

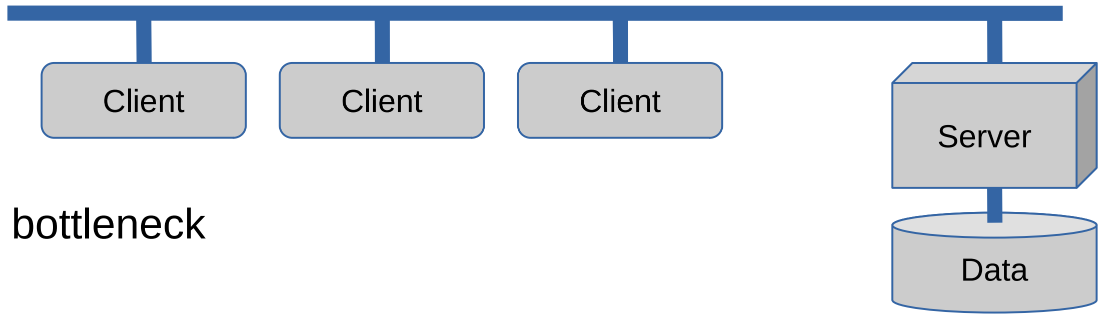
File System Types

Belisama

- Local filesystems
  - Filesystem data only accessible by the owning server
  - Data is placed locally
  - Metadata is maintained locally
  - File locking is done locally
  - Examples: JFS, JFS2, Veritas FS, UFS, ReiserFS, EXT3, EXT4, ..



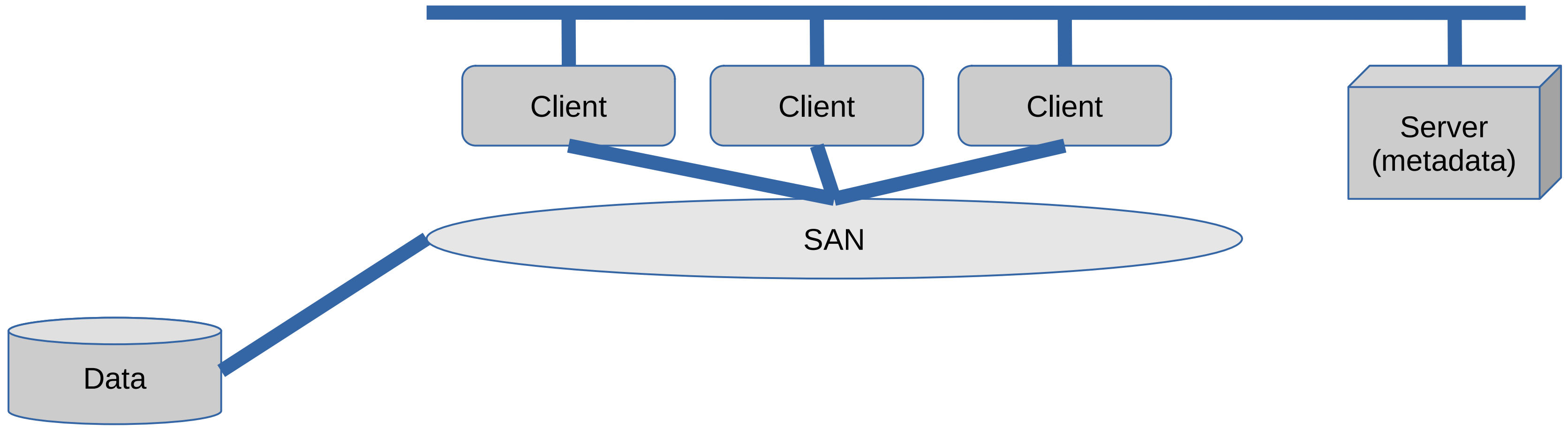
- Remote filesystems
  - Data is placed on remote server
  - Metadata is maintained by remote server
  - File locking is done by remote server
  - Single server might become performance bottleneck
  - Examples: NFS v3/v4 (single server)



File System Types (cont)

Belisama

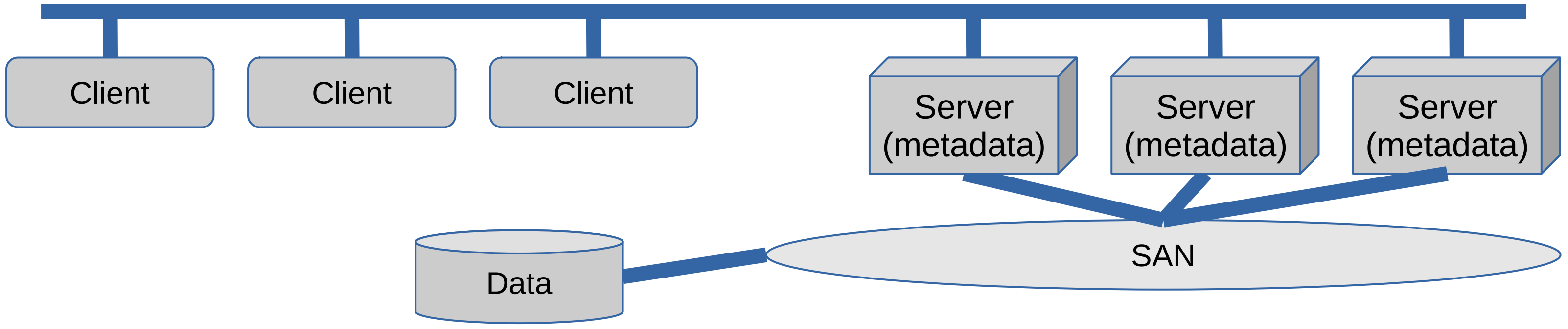
- Shared filesystems
  - Data is placed on shared local disks (e.g. SAN)
  - Metadata is maintained by and stored on a central metadata server
  - File locking is done on the metadata server
  - Metadata server might become performance bottleneck
  - High availability of metadata servers is often limited
  - Examples: SAN file systems; filesystem extensions (e.g. Veritas)



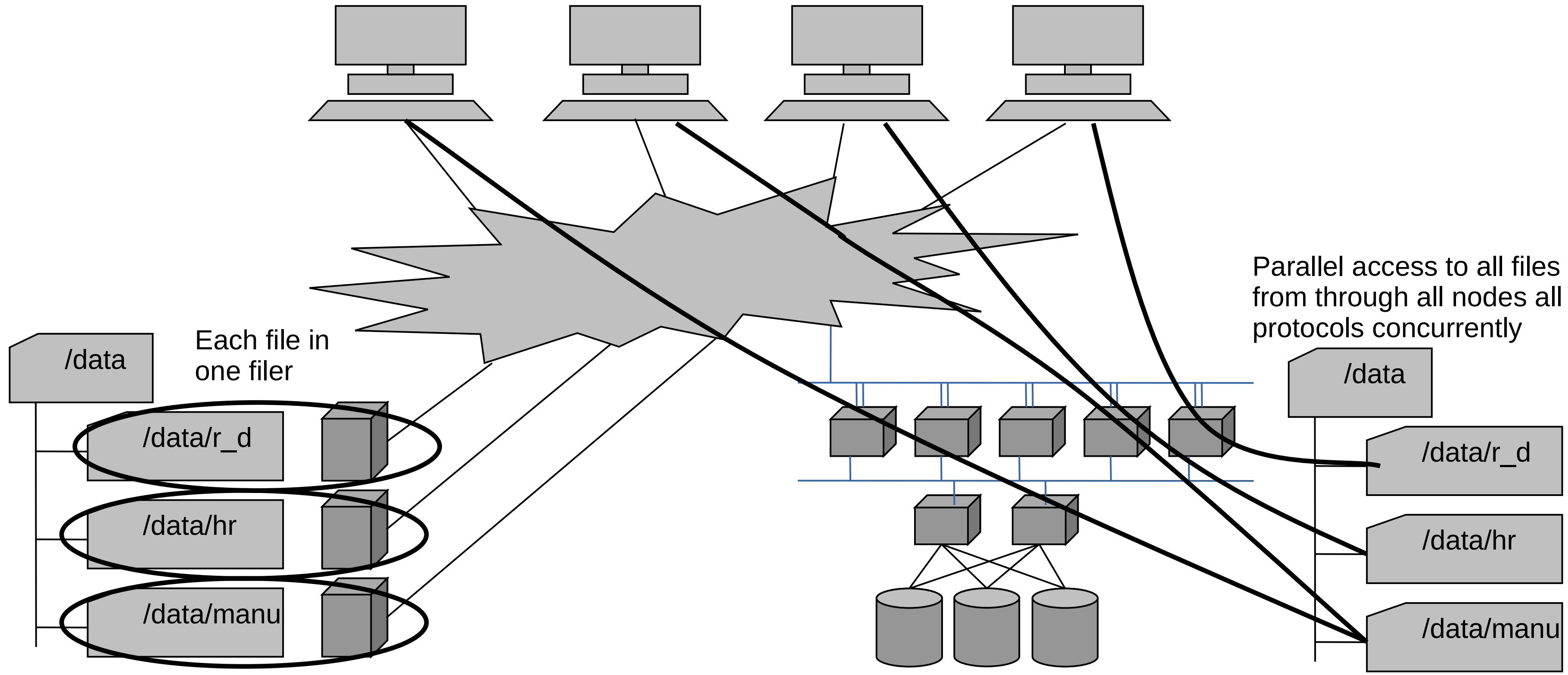
File Systems Types (cont)

Belisama

- Spectrum Scale / General Parallel File System
  - Data is striped across shared local disks (e.g. SAN) or NSD servers
  - Metadata is maintained by all servers in the cluster
  - File locking is distributed across the servers in the cluster
  - Excellent performance and scalability for large amounts of data
  - Very flexible configuration
  - Proven and mature high availability concepts, even for site disaster
  - GPFS Clusters
    - Collection of AIX; Linux; Windows Servers with passwordless ssh communication (or sudo)
    - Manager / non-manager; Quorum / Non-quorum
    - Form a cluster (tie-breaker disks for small clusters)

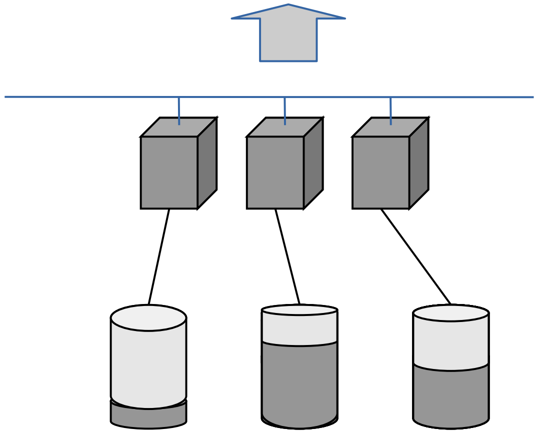


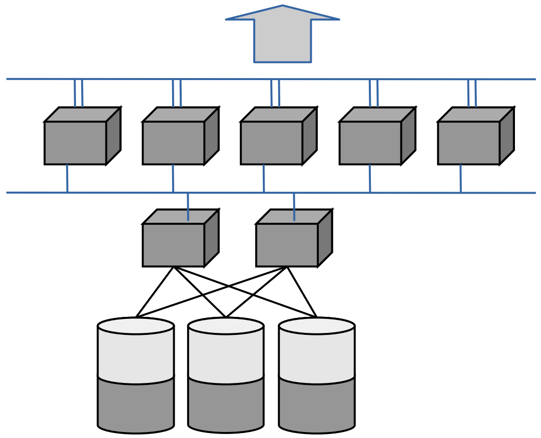
Spectrum Scale – Global Name Space



GPFS compared with Filers

Belisama



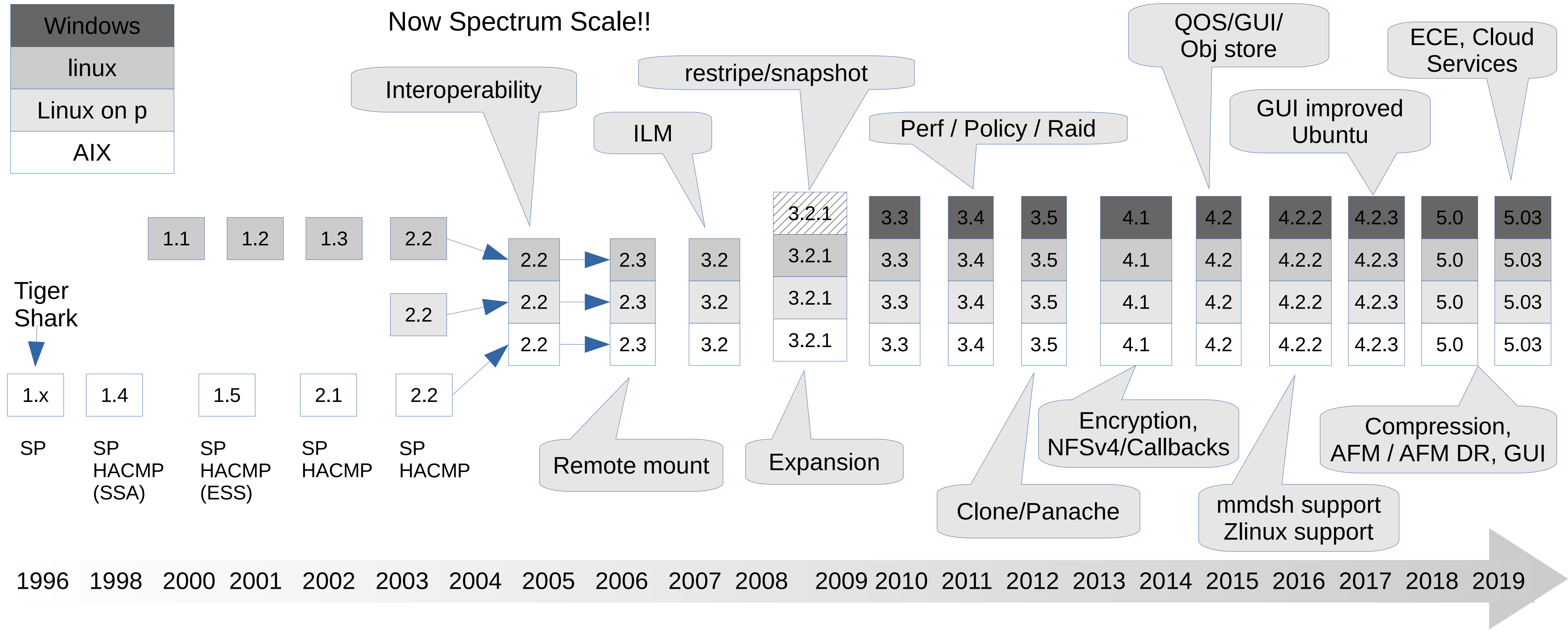


- Classic Fillers
  - “loved my first filler, so easy to manage, but when we installed the 20th….”
  - Individual management
  - Linear cost growth

- GPFS
  - Centrally managed
  - ILM part of GPFS
  - Easy growth and data migration

History and Milestones

Belisama



Usage scenarios

- HPC - Scientific and technical environments
  - Research & HPC
  - Crash & NVH testing, CAE (Automotive and Aerospace)
  - Large Cluster (AIX, Linux, BlueGene/P)
  - WAN Filesystem for Data Grids
- Commercial environments
  - Fast, scalable access to large amounts of file data
  - High Availability clusters (HA)
  - Oracle DB Real Application Clusters (RAC)
  - File System for Data Warehouses (DWH)
  - Media, TV, Medial, Banking and Insurance Customers
  - ESS (SoNAS/SoFS) Samba / CIFS
  - CNFS (Clustered NFS)
  - VTL (Virtual Tape Libraries)
- Systems
  - Blue Gene
  - Mare Nostrum.....
  - Watson
- SAP and oracle certified...
- Spectrum Protect integration (see Advanced admin guide)
- Clustered Network File System (CNFS)
  - See Redbook

PowerVC / novalink Software Defined storage

GPFS Clusters

- 1 to 8,192 Nodes supported
  - Tested up to 5,000 Linux nodes and 2,000 AIX Nodes
  - There are many GPFS installations that contain more than 500 nodes
- Operating Systems include AIX, Linux and Windows
  - AIX 5L; AIX 6.1; AIX 7.1; AIX 7.2
  - pLinux, x_86, x86_64 Distros: RHEL 5, 6 and 7 and SLES 10, 11 and 12 .. Ubuntu
  - Blue Gene (BG/L,BG/P)
  - Windows 2008 Server 64Bit
- Can run a mix of OS levels and a mix of AIX and Linux nodes (and now windows)
- There was a Management GUI 3.2/3.3 – gone in 3.4! But will be back in 4.1 sp2 and ESS

Key strengths

- Mature IBM product generally available since 1998
  - Used by thousands of customers in large production environments
  - Excellent support, FAQ pages, technical forum, papers, ...
  - Constantly introducing enhancements and new features
- Standard, POSIX-compliant UNIX file system interface
  - Buffered I/O, synchronous I/O, asynchronous I/O, Direct I/O
  - Additional non-POSIX extensions (e.g. data-shipping, hints)
- Truly parallel, high performance cluster file system
  - Simultaneous read and write access from different nodes
  - Token-based distributed locking
  - AIX clusters, Linux clusters and even AIX/Linux mixed clusters
  - I/O performance 102 GB/sec with 1.9 PB Storage (ASCII Purple)
  - 2400 GPFS nodes at Mare Nostrum cluster in Barcelona
  - CORAL project (Dept Energy US) ESS 4608 Nodes providing 250PB meeting benchmark of 2.5 TBps in a single stream / creation of 2.6 million 32K files per second.

Key strengths (cont)

Belisama

- Ease of use and robustness
  - Administration can be done from any node with simple commands
  - Online reconfiguration (adding and deleting disks and nodes)
  - High recoverability and increased data availability
  - Information Life-cycle Management (ILM)
- Scalability and performance
  - Scalability to a large numbers of nodes and disks
  - Ability to support extremely large files
  - Striping of data across nodes and disks to maximise throughput
- Flexibility and interoperability
  - Support for mixed clusters running Linux or AIX (sharing disks) plus Windows (not sharing disks)
  - Shared file system access across separate GPFS clusters
  - Improved file serving for Network Filesystem (NFS) v4 functions and performance

Limitations

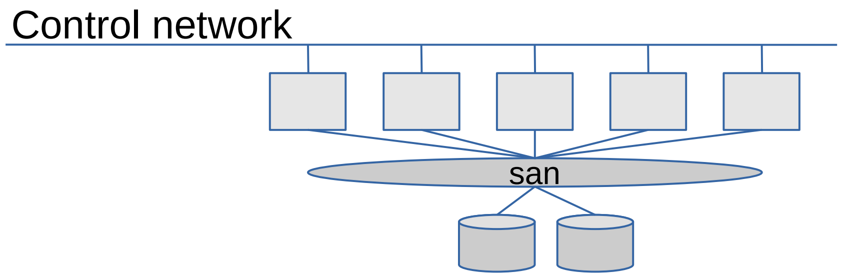
Belisama

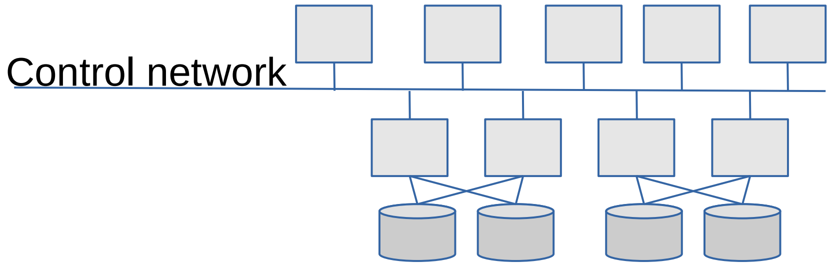
- GPFS 2.3, or later, architectural file system size limit
  - 2^{99} bytes
  - Current tested limit 500PB
- Total number of files per file system
  - 2^{63} (over 40 Billion tested)   Note: GPFS 3.3 and earlier 2 billion
- Total number of nodes 8,192
  - A node is considered in a cluster if:
  - The node shows up in *mmlscluster*,or
  - The node is in a remote cluster and is mounting a file system in the local cluster
- Maximum number of mounted file systems
  - 256
- Maximum disk size
  - Limited by disk device driver and O/S (within constraints of the size if the disks used when filesystem first created)
- Maximum number of snapshots
  - 256

Base Concepts

Belisama

- Technical concepts
  - Shared Disks
    - All data and metadata on globally accessible block storage
  - Wide Striping
    - All data and metadata striped across all disks
    - Files striped block by block across all disks
    - … for throughput and load balancing
  - Distributed Metadata
    - No metadata node – file system nodes manipulate metadata directly
    - Distributed locking coordinates disk access from multiple nodes
    - Metadata updates journalled to shared disk





Principle: scalability through parallelism and autonomy

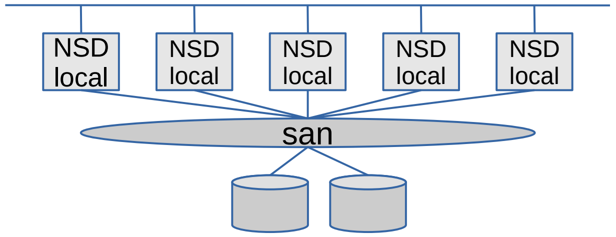
Base Concepts (cont)

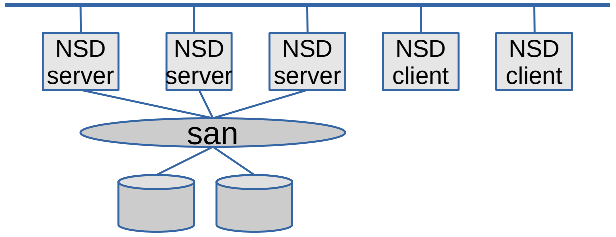
Belisama

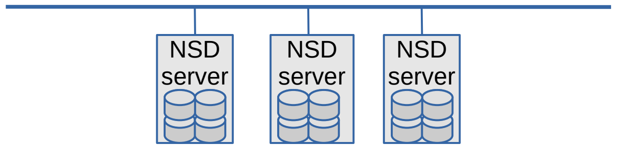
Network shared disks architecture

- Direct attached NSD
  - All nodes are connected to the same Storage Area Network
  - Control information goes over an IP network

- LAN attached NSD
  - Some nodes act as NSD (Network Shared Disk) servers
  - Control information and data goes over an IP network or a high performance switch







Base Concepts (cont)

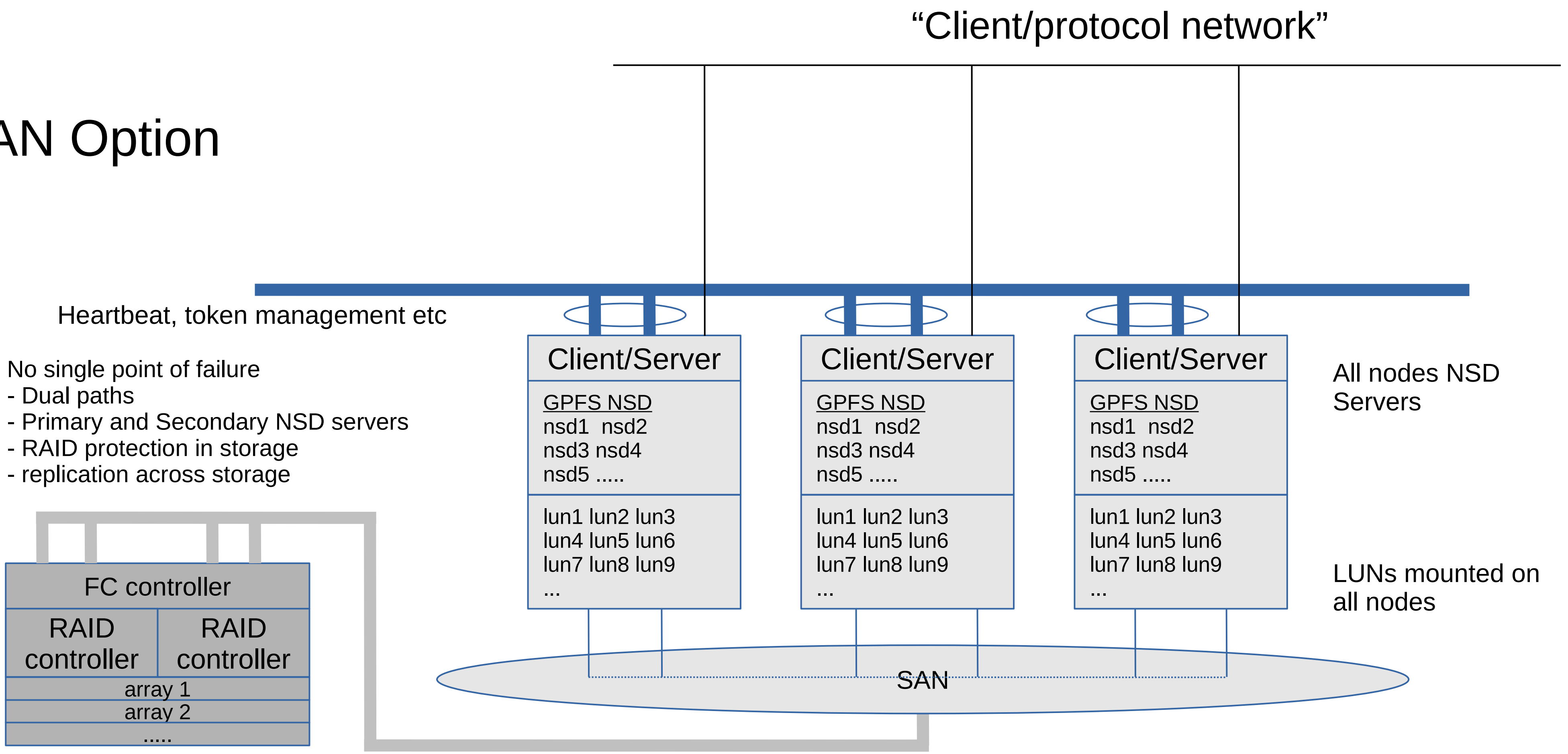
Belisama

- GPFS Components
  - Nodes
    - GPFS clusters consist of AIX nodes, Linux nodes, or a combination thereof.
    - A node is an individual operating system image within a cluster, either on a single computer or on a system partition.
  - Shared network
    - A TCP/IP network used for the communication between GPFS daemons
    - Can also be used for transferring data from and to the NSDs
  - Network shared disks (NSDs)
    - All disks utilised by GPFS must first be given a globally accessible NSD name
    - NSD provide a method for cluster-wide disk naming and access (all nodes see /dev/nsd_00x)
    - On Linux machines running GPFS, you may give an NSD name to:
      - Physical disks
      - Logical partitions of a disk
      - Representations of physical disks (such as LUNs)
    - On AIX machines running GPFS, you may give an NSD name to:
      - Physical disks
      - Virtual shared disks (old)
      - Representations of physical disks (such as LUNs)

Base Concepts (cont)

Belisama

SAN Option



No single point of failure
- Dual paths
- Primary and Secondary NSD servers
- RAID protection in storage
- replication across storage

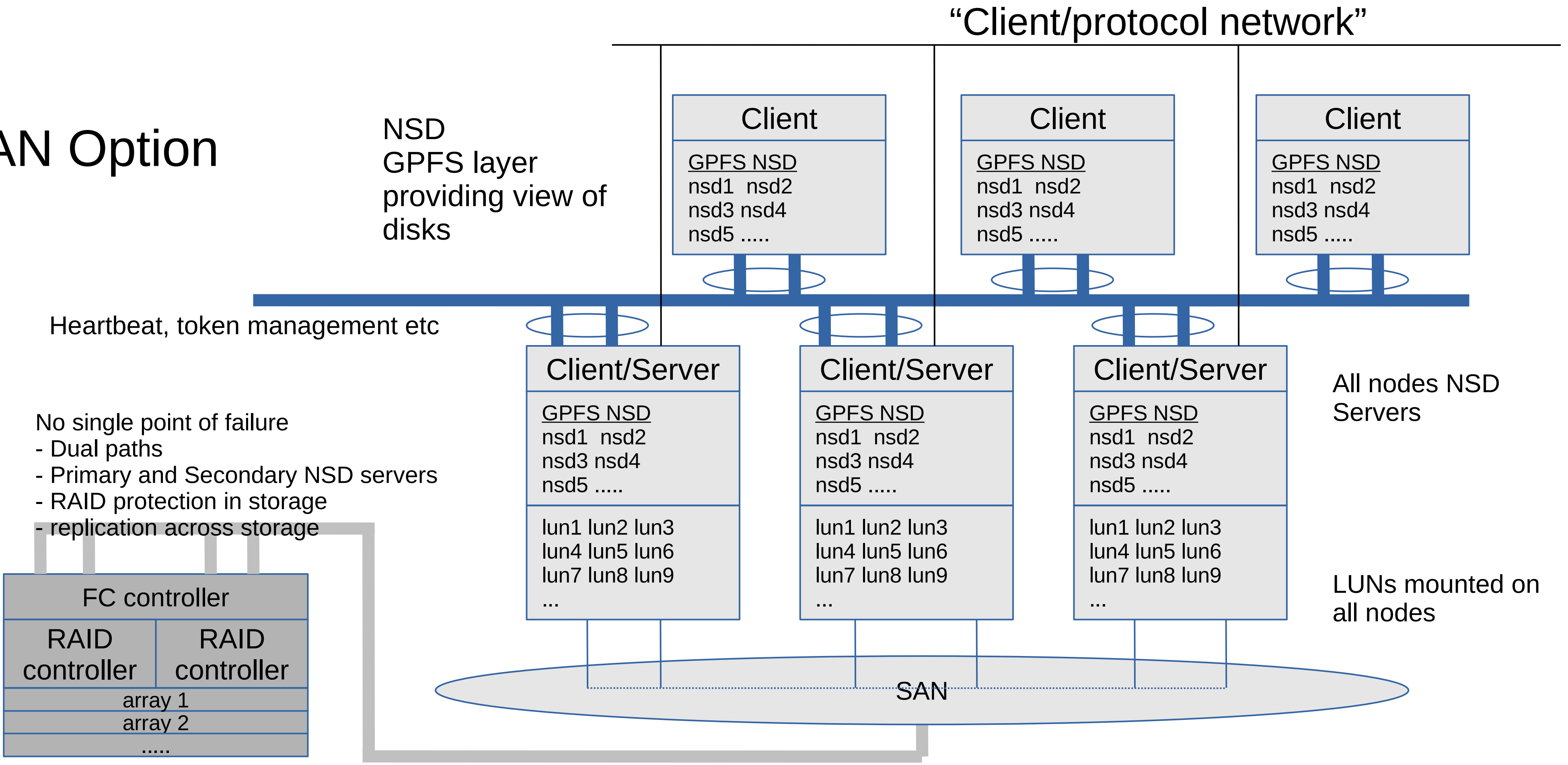
Base Concepts (cont)

Belisama

SAN Option

NSD
GPFS layer providing view of disks

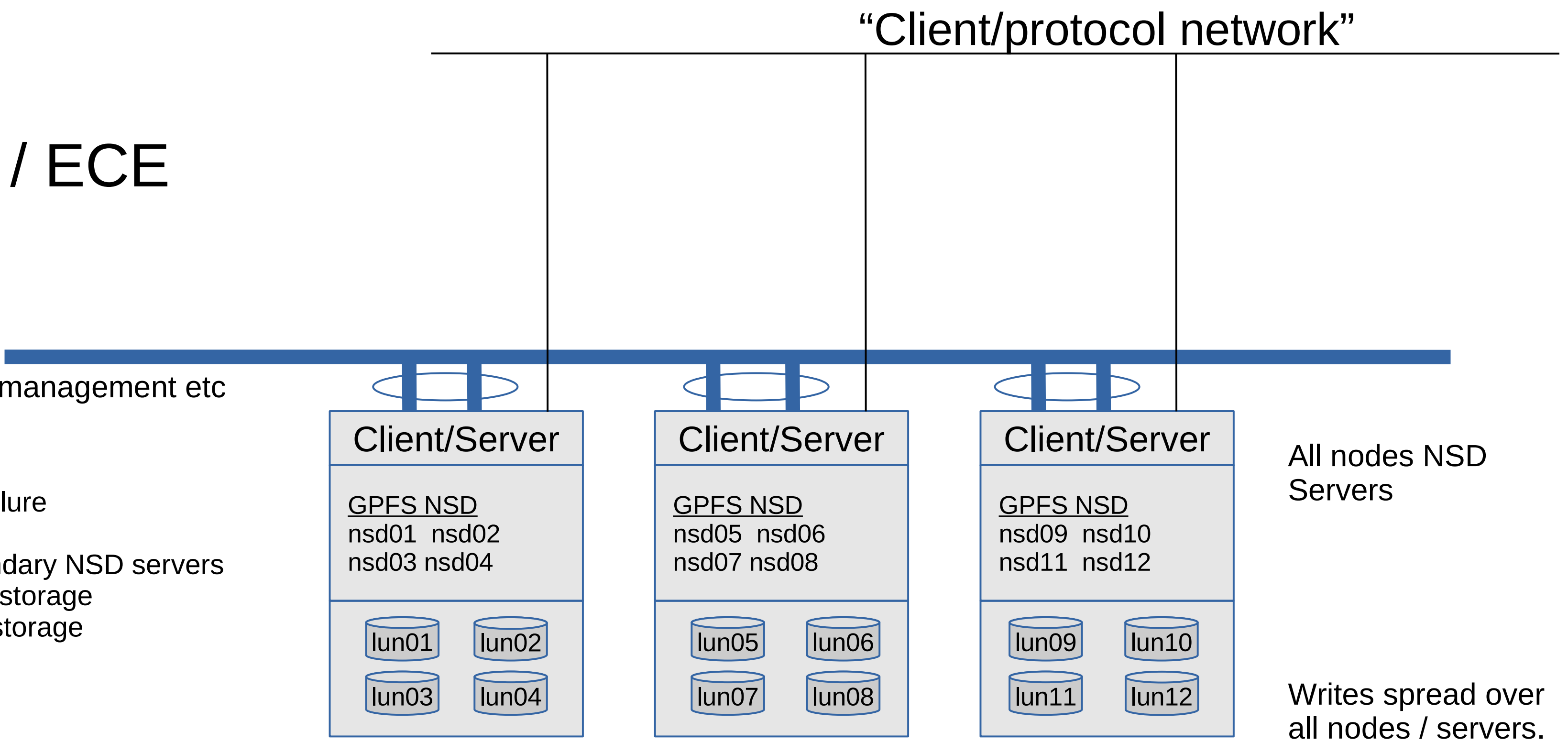
No single point of failure
- Dual paths
- Primary and Secondary NSD servers
- RAID protection in storage
- replication across storage



Base Concepts (cont)

Belisama

FPO Option / ECE



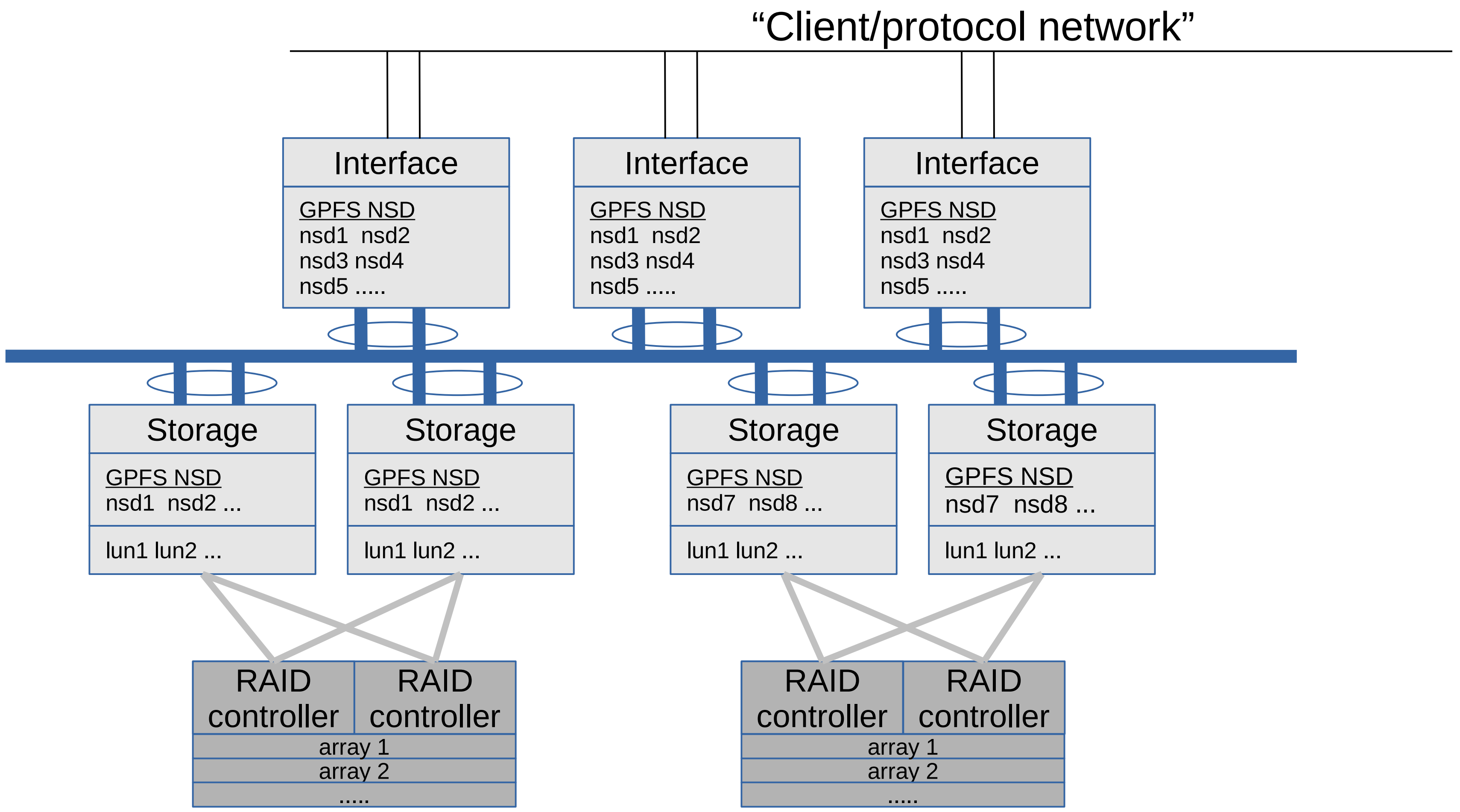
No single point of failure
- Dual paths
- Primary and Secondary NSD servers
- RAID protection in storage
- replication across storage

All nodes NSD Servers

Writes spread over all nodes / servers. Replicated and placement optimised by proximity to the process

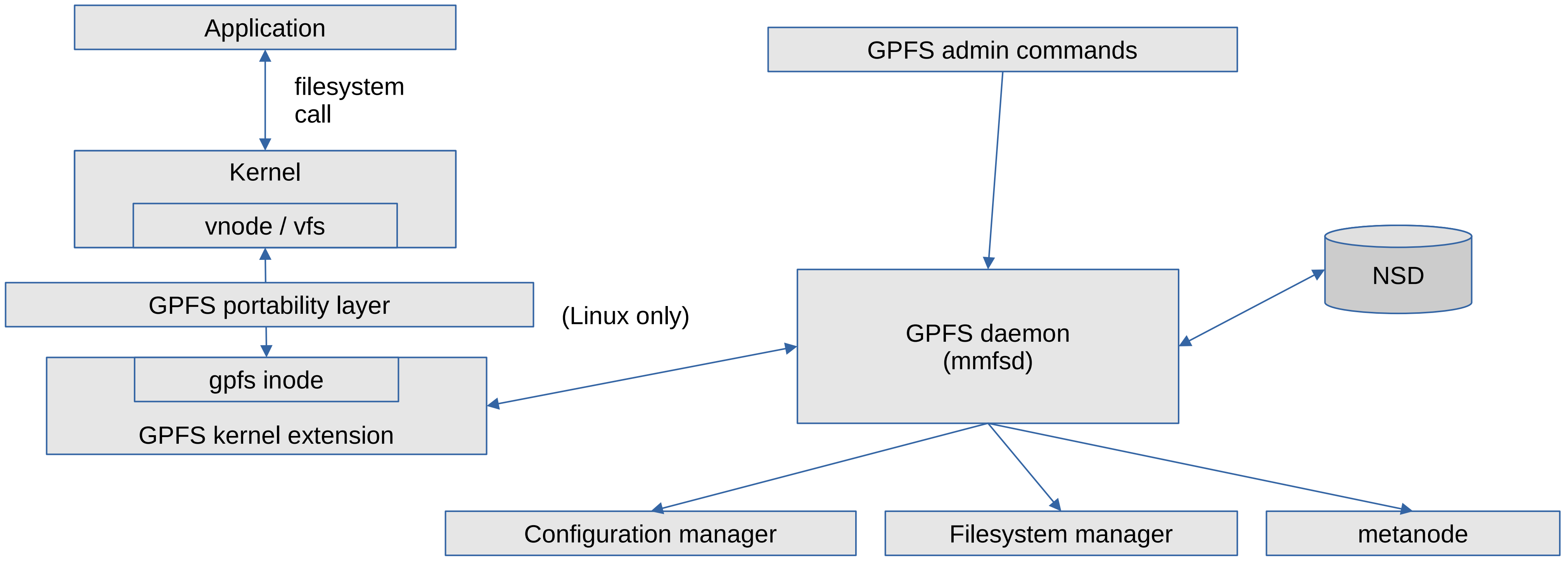
Base Concepts (cont) - the ESS view of GPFS

Belisama



Base Concepts (cont) – the GPFS structure

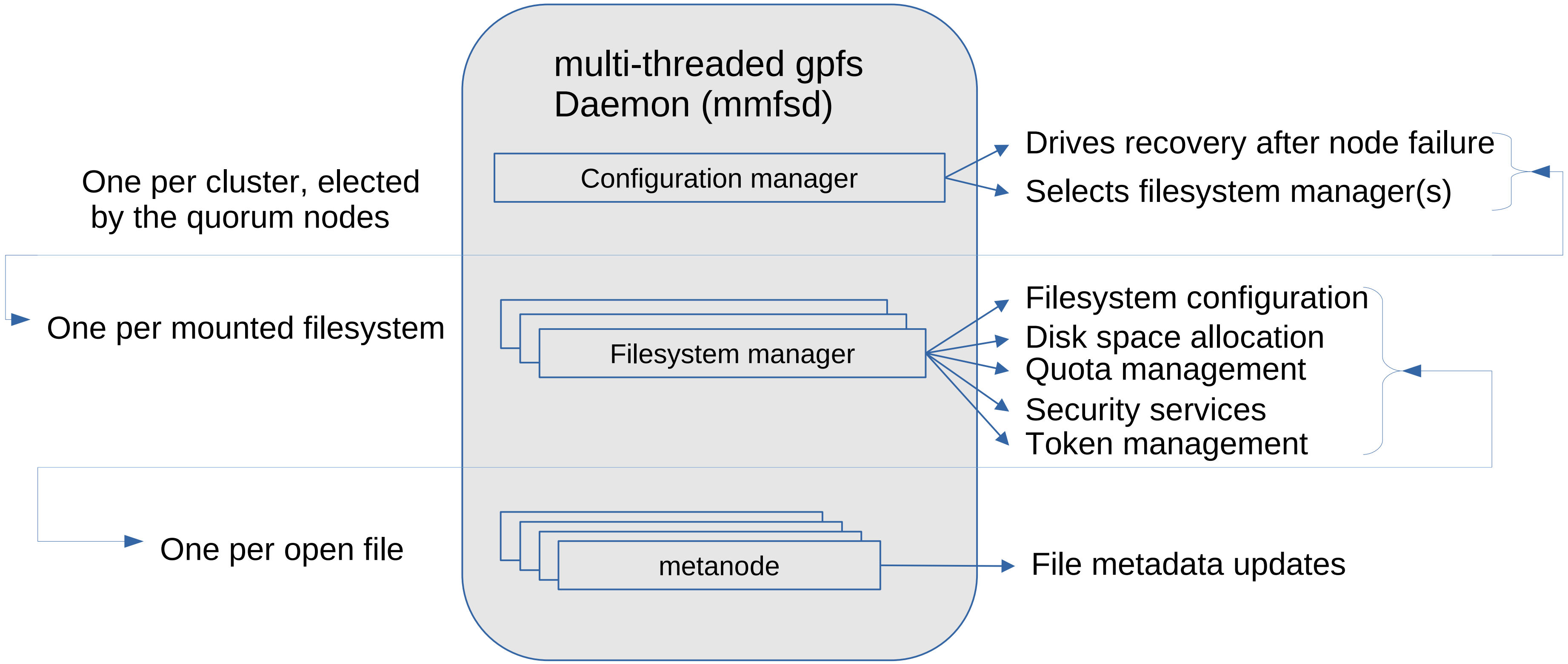
Belisama



Base Concepts (cont)

Belisama

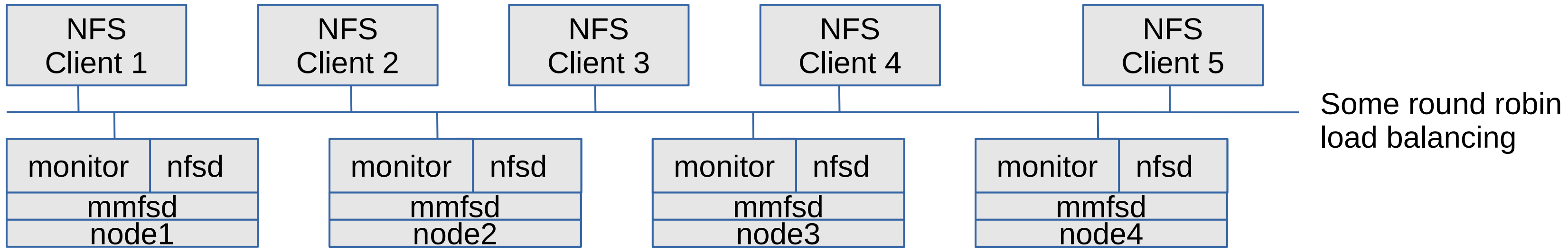
- GPFS daemon (mmfsd) roles



Base Concepts (cont)

Belisama

- HA-NFS / cNFS
  - GPFS and HA-NFS features contained entirely in the GPFS cluster
  - Clients access storage via NFS
    - Clients use vanilla NFS (no special software – only DNS RR)
    - Clients can be AIX, Linux, Solaris, MAC (or any other Unix based OS)



Plan a GPFS cluster

- Plan a GPFS cluster
  - Plan hardware
    - Supported
    - Storage and zoning
    - Firmware
  - Operating system
    - Supported
    - Fixes
  - GPFS
    - Install and update
  - Linux
    - Compile the General portability layer
  - Planning networks
    - Open firewall ports for GPFS daemon; ssh; ping
- Create a node definition file
  - Exchange keys and ensure that ssh is passwordless for root from every node to every other node (including itself)
- Create a GPFS cluster

Planning considerations

- Nodes
  - Sizing
  - Number of nodes to provide thoughput
  - Node quorum considerations (or tiebreaker disks for very small clusters)
    - Small odd number of most reliable nodes spread across the infrastructure
- Filesystems
  - Number of filesystems
    - Different than planning local file systems, fewer file systems often better
    - Multiple applications can often share a file system
    - Split clusters – separate security contexts but can cross mount
    - Are the requirements within architectural limits?
      - < 2^{63} files, with Size limit 2^{99} Bytes -> Current tested size order for 100s PB
    - Create file systems to support performance requirements
      - Disk type differences can be addressed using Storage Pools
  - Block size
    - 64KB blocksize for small block random I/O
      - Examples: Email service, Web applications
    - 256KB for standard file service and for larger block random IO (anything bigger very expensive for small writes)
      - Examples: User file service, Grid analytical systems
    - 2MB to 4MB for large block sequential read/write
      - Examples: Digital media, Data warehousing, Weather modelling
    - Match filesystems to application blocksize:
      - /gpfs1 – 64K block size
      - /gpfs2 – 256K block size
  - Numbers of filesystem replicas

Further planning considerations

- Metadata
  - Access required when change is made to files “metadata”
  - Manger nodes in cluster can manage metadata
  - Can be bottleneck with 1000s files created / deleted.
  - Use smaller number of filesystems
  - Each application has own sub directory
  - Use different directories to reduce contention
  - Since 4.1, you can set the inode size (default is 4KB)
- Working in a multi-node environment
  - Keep consistent
    - applications
    - user data
    - patches
  - Understand what will happen if two applications open the one file, both to the file and that it is expensive locking
  - Some operations in a parallel environment are not cheap – for example stat(2) and readdir(3).
  - Frequent dir scans looking for new files will hurt performance
  - stat() on a file from another node will affect your write performance
- Segments = 1/32 of a block
  - The smallest amount of disk allocated by GPFS (variable block size introduced in 5)

Still further planning considerations

Belisama

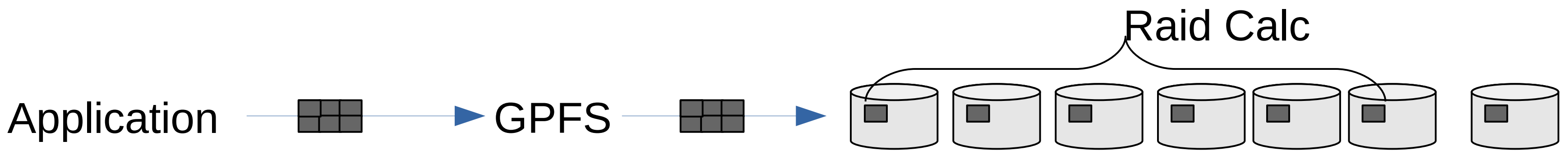
- Application - number of filesystems
  - Different than planning local file systems
  - Fewer file systems
  - Multiple applications can often share a file system
  - Split clusters – separate security contexts but can cross mount
  - Are the requirements within architectural limits?
    - < 4 Billion Files (tested 3.4)
    - Size limit 2^{99} Bytes -> Current tested size order for 100s PB
  - Create file systems to support performance requirements
    - Disk type differences can be addressed using Storage Pools
  - Application is GPFS aware – use the GPFS API
- Direct I/O caching option
  - The Direct I/O caching policy bypasses file cache and transfers data directly from disk into the user space buffer, as opposed to using the normal cache policy of placing pages in kernel memory. Applications with poor cache hit rates or very large I/Os may benefit from the use of Direct I/O
    - mmchattr -D [{yes | no }] filename
  - Or
    - Direct I/O may also be specified by supplying the O_DIRECT file access mode on the open() of the file.
  - mmchattr can also be used to set files
    - Set files as immutable
    - Set files as append only
    - Number of replicas of data and/or metadata
    - Storage pool

Storage considerations

Belisama

- Data storage
  - Two types of data storage
    - Metadata (inode)
    - File Data
  - Metadata
    - File stat info: Date created, last access time, size
    - Reserved files (inode file, allocation map, inode map)
    - Indirect blocks, directories, symbolic links
    - Active Policy definitions
  - File Data
    - Contents of the file(s)
  - Metadata and Data can be shared or separate

- Data storage (cont)
  - Arrays are defined as metadataOnly, dataOnly or dataAndMetadata
  - Shared metadata and data
    - Works well for many applications
  - Separate data and metadata
    - Using Storage pools
    - Metadata can only be stored in the system storage pool
    - Reduce contention on metadata
    - Reduce storage costs – Isolate metadata on Fibre, data on SATA
  - tune LUN config for access type / access size



Storage considerations (cont)

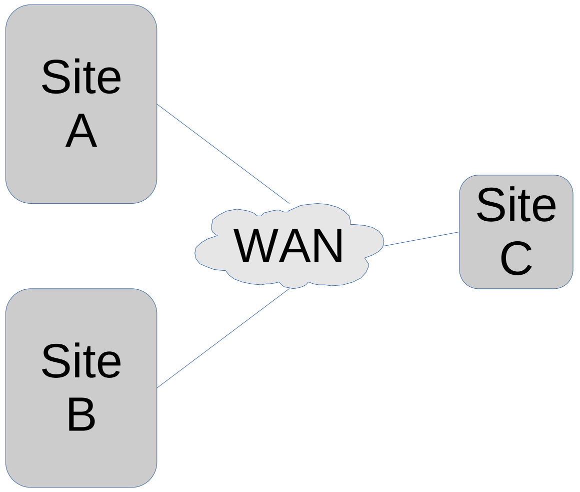
- Storage Configuration
  - IO Balance
  - Even number of LUNS per path
  - Even number of LUNS per NSD server
- RAID Level
  - Right level for IO pattern
- Cache Configuration
  - Read ahead not recommended for –j scatter
  - Write cache for metadata LUNS
- Interconnect Selection and Tuning
  - TCP/IP Tuning
  - Fibre Channel Settings
  - InfiniBand Configuration

Network considerations

- Network
  - Firewall settings
  - Availability and throughput - use Bonding
  - Enable jumbo frames if supported by the switch
  - Don't use DNS for GPFS private network (netsvc.conf or nsswitch.conf -> hosts)
  - For security use ssh/scp/sftp, turn off unnecessary services, password rules and expiry,
  - Keep user information consistent in clustered environment

GPFS Site considerations

- Availability / Multi site
  - Distributed Data
  - data is distributed across 2 sites, 3rd site contains quorum node for availability
  - Sites A and B
    - contain the core GPFS nodes and storage
    - Multiple quorum nodes in each site
  - Site C
    - “Laptop solution”
    - contains a single quorum node, filesystem descriptor
    - Serves as tie breaker if one of the other sites becomes inaccessible



Inside GPFS

Belisama

- A GPFS cluster
- Nodes, disks and filesystems
- Using the filesystem
- Monitoring the cluster

Get Started / How to Guide

Belisama

- Creating a GPFS Cluster
  - Plan a GPFS cluster
  - Create a node definition file
  - Create a GPFS cluster
  - View information on the GPFS cluster
  - View information on the GPFS configuration
  - Startup GPFS on the nodes
  - View information on the status of the GPFS cluster
  - Stop GPFS on the nodes

Create a node definition file

Belisama

- Create a file with one node descriptor line per node
  - NodeName:NodeDesignations:AdminNodeName
  - Where:
    - NodeName is either IP address or IP name of the interface that GPFS should use to communicate with the other nodes
    - NodeDesignations is an optional “-” separated list of node roles (quorum or nonquorum, manager or client)
    - AdminNodeName is an optional IP address or IP name, that GPFS should use for administrative commands instead of NodeName

```
For example:  /tmp/gpfs-nodes.txt
      node1:quorum-manager:n1
      node2:quorum-manager:n2
      node3:quorum-manager:n3
      node4:nonquorum-manager:n4
```

Create a GPFS cluster

Belisama

- The most important options of the mmcrcluster command are:
  - -A: Startup GPFS daemons automatically when nodes come up.
  - -N <NodeDefFile>: specifies the node definition file (list of node descriptors)
  - --ccr-enable: Enables the configuration server repository to store redundant copies of the configuration data files on all quorum nodes (default)
  - --ccr-disable: The previous configuration with a primary and secondary configuration server node
    - -p <PrimaryServer>: specifies the primary cluster configuration server node
    - -s <SecondarySrv>: Specifies the secondary cluster configuration server node
  - -R <RemoteFileCopy>: path/name for remote copy program, e.g. /usr/bin/scp
  - -r <RemoteShellCmd>: path/name for remote shell program, e.g. /usr/bin/ssh

```
#  mmcrcluster -N /tmp/gpfs-nodes.txt -p node2 -s node3 -r /usr/bin/ssh -R /usr/bin/scp
Wed Jun 24 18:34:26 EET 2009: mmcrcluster: Processing node node1
Wed Jun 24 18:34:27 EET 2009: mmcrcluster: Processing node node2
Wed Jun 24 18:34:28 EET 2009: mmcrcluster: Processing node node3
Wed Jun 24 18:34:30 EET 2009: mmcrcluster: Processing node node4
mmcrcluster: Command successfully completed
mmcrcluster: Propagating the cluster configuration data to all   affected nodes.  This is an asynchronous process.
```

View information on the GPFS cluster

Belisama

- The mmlscluster command displays information on the cluster configuration, NOT the status of the cluster
  - Information about the cluster itself, such as cluster name, remote shell / remote copy command and cluster configuration servers
  - Information about the nodes in the cluster, such as IP address and node designation

```
# mmlscluster
GPFS cluster information
========================
  GPFS cluster name:         ess_test1
  GPFS cluster id:           12398410922139748073
  GPFS UID domain:           ess_test1.syd-demo.ibm
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp

GPFS cluster configuration servers:
-----------------------------------
  Primary server:    ts1.red.com
  Secondary server:  ts2.red.com

 Node  Daemon node name            IP address       Admin node name       Designation
-----------------------------------------------------------------------------------------
   1   ts1.red.com                 172.16.1.11      ts1.red.com           quorum-manager
   2   ts2.red.com                 172.16.1.12      ts2.red.com           quorum-manager
   3   ts3.red.com                 172.16.1.13      ts3.red.com           quorum-manager
```

View information on the GPFS configuration

Belisama

- The mmlsconfig command displays information on the GPFS configuration parameters and file systems
  - The first section shows global GPFS configuration parameters
    - Parameters that are unique to this GPFS cluster such as the name
    - Parameters that do not have the default value
    - At the end of this section there might be the node name in brackets, followed by individual parameter settings for this node
  - A list of file systems defined in this GPFS cluster

```
# mmlsconfig
[root@ts1 ras]# mmlsconfig
Configuration data for cluster sofs151.red.com:
-----------------------------------------------
clusterName ess_test1.syd-demo.ibm
clusterId 12398410922139748073
clusterType lc
autoload yes
MinReleaseLevel 4.1.0.1
dmapiFileHandleSize 32
leaseRecoveryWait 3
...
```

```
# ..... (cont)
maxFilesToCache 20000
maxStatCache  80000
FailureDetectionTime 10
maxMBpS 500
unmountOnDiskFail no
allowSambaCaseInsensitiveLookup no
enableLowspaceEvents yes
cipherList AUTHONLY
pagepool 64M
dmapiDataEventRetry 2
verifyGpfsReady yes
```

Important tuning parameters

Belisama

- GPFS use of memory
  - Two areas of memory
  - Pinned (pagepool) – used to store user data and filesystem metadata to support I/O operations
  - Not Pinned – two levels of cache for storing file metadata
  - Pagepool
    - The pagepool mechanism allows GPFS to implement read as well as write requests asynchronously. Increasing the size of pagepool increases the amount of data or metadata that GPFS can cache without requiring synchronous I/O. The amount of memory available for GPFS pagepool on a particular node may be restricted by the operating system and other software running on the node.
    - The following types of I/O may benefit from increasing the pagepool:
      - There are frequent writes that can be overlapped with application execution.
      - There is frequent reuse of file data that can fit in the pagepool.
      - The I/O pattern contains various sequential reads large enough that the prefetching data improves performance.
    - For NSD Servers, 3*#LUNS*maxBlockSize should be $\leq$ 30% pagepool
  - maxFilesToCache
    - This space needs to be big enough for currently opened files and to cache some recently used files (default 1000). If there are applications that test files, without actually opening them – such as backups, this value may be increased.
    - Memory used is maxFilesToCache * 3KB (2.5 pre 3.3)

Important tuning parameters (cont)

- Memory (cont)
  - maxStatCache
    - This parameter sets aside additional pageable memory to cache attributes of files that are not currently in the regular file cache (default is 4000). This is useful to improve the performance of both the system and GPFS stat( ) calls for applications with a working set that does not fit in the regular file cache.
    - maxStatCache × 400 bytes (176 pre 3.3)
  - The total amount of memory GPFS uses to cache file data and metadata is arrived at by adding pagepool to the amount of memory required to hold inodes and control data structures (maxFilesToCache × 3 KB), and the memory for the stat cache (maxStatCache × 400 bytes) together.
  - The combined amount of memory to hold inodes, control data structures, and the stat cache is limited to 50% of the physical memory on a node running GPFS.
- ShareMemLimit (increased in 3.2)
  - Size of the shared memory segment (kernel and mmfs daemon) used by GPFS

Important tuning parameters (cont)

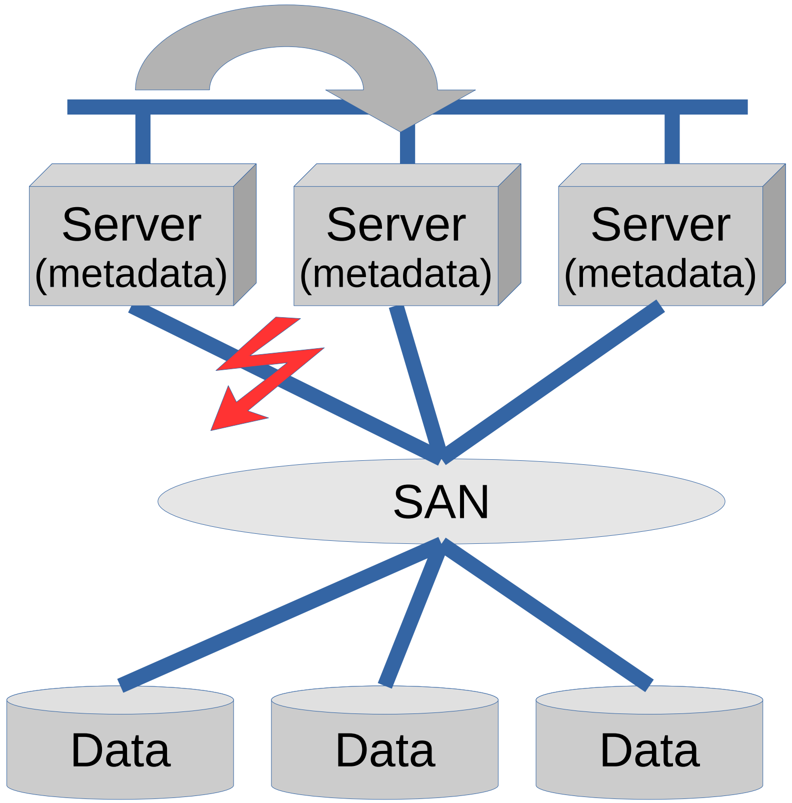
Belisama

- maxMBps
  - This value is usually set to be two times the maximum I/O throughput that GPFS can achieve. Not used by the NSD Servers, only application nodes doing sequential access.
- Number nodes to mount
  - GPFS uses for internal tuning (default 32)
- Exact mtime, suppress atime
  - So stat() calls are accurate – but expensive!

Important tuning parameters (cont)

Belisama

- Define NSD Servers
  - In GPFS 3.2.1 and above you can define up to 8 NSD Servers for each NSD
  - If the path to the disks for a node fails, and other NSD server are set, then the node will continue to operate, communicating with the remaining NSD Server(s) by the GPFS private network. The customer needs to decide whether they want to have the nodes always serving the filesystem (and therefore running their application) at the expense of increased network traffic. The alternative is to set all as "directly attached".
  - Define multiple NSD servers for each NSD.



Important tuning parameters (cont)

- distributedTokenService
  - Specifies whether the token server role for a file system should be limited to only the file system manager node (no), or distributed to other nodes for better file system performance (yes) – default is yes.
- Exact mtime mount
  - if yes (the default) them mtime and ctime will always be correct for the stat() call. If no, can be out for a couple of minutes.
  - Recommend: If the Application Vendor has no concerns, set to no
- Suppress atime mount
  - atime represents the time when the file was last accessed. This parameter controls the updating of the atime value. The default it is no, which results in updating atime locally in memory whenever a file is read, but the value is not visible to other nodes until after the file is closed. If an accurate atime is needed, set to no, the default.
  - Recommend: If the Application Vendor has no concerns, set to yes

Important tuning parameters (cont)

Belisama

- Prior to GPFS 4.2.0.3 we used to tune worker1threads and worker3 threads
  - worker1threads is the total number of concurrent application requests that can be processed at one time. This may include metadata operations like file stat() requests, open or close and for data operations.
  - worker3threads specifies the number of threads to use for inode prefetch.
  - Typically these values were set at their default then increased after reviewing cluster operation and mmdiag output.
- workerthreads were documented in 4.2.1 – GPFS will tune on configuration on startup. Tune as did before with worker1threads (for example set to 512 for high performance NSD server clusters)
- The default inode size since 4.1 is 4KB

Starting the cluster

Belisama

- mmstartup starts the GPFS subsystem
  - -N Nodelist to start the cluster on one or a subset of the nodes
  - The –a option starts GPFS on all nodes

```
# mmstartup -a
SUn Jan 28 11:54:33 EST 2018: mmstartup: Starting GPFS ...
```

Viewing the state of the cluster

- The mmgetstate command displays the state of the GPFS daemon on one or more nodes
  - -a shows the status of GPFS on all nodes
  - -L shows extended node information
  - -s shows a summary status

```
# mmgetstate -aL

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
-------------------------------------------------------------------------------------
       1      ts1                2        0          3       active       quorum node
       2      ts2                2        0          3       active       quorum node
       3      ts3                2        0          3       arbitrating  quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:        3
Number of local nodes active in the cluster:   3
Number of remote nodes joined in this cluster: 0
Number of quorum nodes defined in the cluster: 3
Number of quorum nodes active in the cluster:  2
Quorum = 2, Quorum achieved
```

Stopping the cluster

Belisama

- mmshutdown unmounts the GPFS file systems and stops the daemon on a node or nodes
  - -N nodelist stops on a node or subset of nodes.
  - -a stops on all nodes

```
# mmshutdown  -a
Sun Jan 28 11:51:55 EST 2018: mmshutdown: Starting force unmount of GPFS file systems
Sun Jan 28 11:52:18 EST 2018: mmshutdown: Shutting down GPFS daemons
ts2.red.com:  Shutting down!
ts3.red.com:  Shutting down!
ts1.red.com:  Shutting down!
ts2.red.com:  'shutdown' command about to kill process 3056
ts2.red.com:  Unloading modules from /usr/lpp/mmfs/bin
ts2.red.com:  Unloading module mmfs
ts3.red.com:  'shutdown' command about to kill process 3312
ts3.red.com:  Unloading modules from /usr/lpp/mmfs/bin
ts2.red.com:  Unloading module mmfslinux
ts3.red.com:  Unloading module mmfs
ts2.red.com:  Unloading module tracedev
ts3.red.com:  Unloading module mmfslinux
ts3.red.com:  Unloading module tracedev
ts1.red.com:  'shutdown' command about to kill process 3588
ts1.red.com:  Unloading modules from /usr/lpp/mmfs/bin
ts1.red.com:  Unloading module mmfs
ts1.red.com:  Unloading module mmfslinux
ts1.red.com:  Unloading module tracedev
Sun Jan 28 11:52:45 EST 2018: mmshutdown: Finished
```

Get Started / How to Guide (cont)

- Network Shared Disk infrastructure
  - Create a NSD descriptor file for direct attached NSD
  - Create a NSD descriptor file for NSD over LAN
  - Create network shared disks
  - Create a GPFS filesystem

Create a NSD

Belisama

- Disks for use with GPFS need to be defined and formatted, this is done by the mmcrnsd command.
- This command requires input in form of a NSD descriptor file
- Each disk is specified in one stanza with the following format:

```
%nsd:
nsd=NsdName
usage={dataOnly | metadataOnly |
       dataAndMetadata | descOnly}
failureGroup=FailureGroup
pool=StoragePool
servers=ServerList
device=DiskName
```

- Where

  The only required entry is DiskName, which is the block device name for the disk appearing in /dev

  - You may omit the other entries, but using the old format you have to put all colons in the line, even if the column itself is empty
- Once mmcrnsd has completed
  - the NSDs are usable in GPFS
  - In older versions the NSD descriptor file was updated, so that it can be used as input file for other commands

NSD descriptor file

Belisama

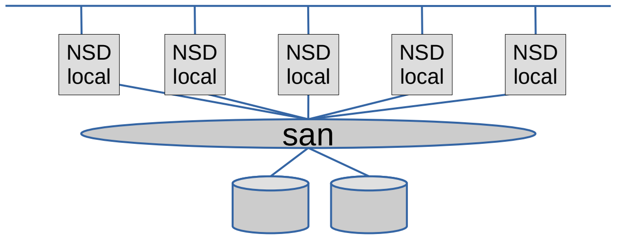
- Records in the NDS descriptor file are:
  - DiskName
    - the block device name appearing in /dev for the disk
  - NSDServer (up to 8, “,” separated)
    - the name of the primary NSD server node. If empty, the disk is assumed to be SAN-attached to all nodes
  - No longer used (was backup nsd server)
  - DiskUsage
    - What kind of information should be stored on this NSD
      - dataAndMetadata (the default)
      - dataOnly Indicates that the disk contains data and no metadata
      - metadataOnly Indicates that the disk contains metadata only
      - descOnly can contain a copy of the file system descriptor only
  - FailureGroup
    - A number identifying the failure group. This concept is explained later when discussing high availability
  - DesiredName
    - Specify the name for the NSD to be created (default gpfsNNnsd)
  - StoragePool
    - Specifies the name of the storage pool that the NSD is assigned to and is used to group like disks for ILM.

```
%nsd:
nsd=NsdName
usage={dataOnly | metadataOnly |
       dataAndMetadata | descOnly}
failureGroup=FailureGroup
pool=StoragePool
servers=ServerList
device=DiskName
```

Create a NSD descriptor file for direct attached NSD

- The DiskName has to be set to the name of the block device in /dev, as it appears on the node where the mmcrnsd command will run
  - It is possible, that the same disk will have different names on different nodes. GPFS identifies this automatically when running mmcrnsd and updates it’s internal configuration accordingly
  - You can leave server list empty, and only SAN access is supported, if you specify the server list, then the NSD Server can be a NSD client if there is a SAN error, but local access will take precedence.

```
%nsd:
nsd=nsd01
usage=dataAndMetadata
failureGroup=-1
pool=System
device=hdisk6
%nsd:
nsd=nsd02
usage=dataAndMetadata
failureGroup=-1
pool=System
device=hdisk7
```

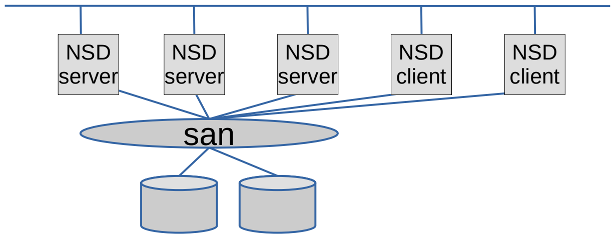


Create a NSD descriptor file for NSD over LAN

Belisama

- The DiskName has to be set to the name of the block device in /dev, as it appears on the PrimaryServer node
- You have to fill the server list (up to 8 servers)
  - It is highly recommended to have more than one server available in case the first server fails.

```
%nsd:
nsd=nsd01
usage=dataAndMetadata
failureGroup=-1
pool=System
servers=Node1, Node2, Node3
device=hdisk6
%nsd:
nsd=nsd02
usage=dataAndMetadata
failureGroup=-1
pool=System
servers=Node2, Node3, Node1
device=hdisk7
```



Create network shared disks

- mmcrnsd creates and formats disks
  - -F: Specify NSD descriptor file
  - -v only format blank disks
- mmlsnsd lists defined disks and usage
- mmcrnsd changes the NSD descriptor file for later use

```
# mmcrnsd -F diskdesc.txt
mmcrnsd: Processing disk hdisk6
mmcrnsd: Processing disk hdisk7
mmcrnsd: Propagating the cluster information to all affected nodes.  This is an asynchronous process.

# mmmlsnsd
 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 (free disk)   nsd01        (directly attached)
 (free disk)   nsd02        (directly attached)
```

mmlsd information

Belisama

```
# mmmlsnsd
 File system   Disk name    NSD servers
---------------------------------------------------------------------------
 gpfs0         nsd01        (directly attached)
 gpfs0         nsd02        (directly attached)
```

```
# mmlsnsd -X

 Disk name    NSD volume ID      Device         Devtype  Node name                Remarks
---------------------------------------------------------------------------------------------
 nsd01        AC10010B49768D18   /dev/hdisk6    hdisk    ts1
 nsd02        AC10010B49768D48   /dev/hdisk7    hdisk    ts1
```

Create a GPFS filesystem

Belisama

- Note: With the old version of the NSD text file, the mmcrnsd command used to modifiy the file so that it could then be used to create the filesystem. However with the new stanza format, the same file can be used to create NSDs and the filesystem. Remember that all NSDs in the file will be used.

```
%nsd:
nsd=nsd01
usage=dataAndMetadata
failureGroup=-1
pool=System
servers=Node1, Node2, Node3
device=hdisk6
%nsd:
nsd=nsd02
usage=dataAndMetadata
failureGroup=-1
pool=System
servers=Node2, Node3, Node1
device=hdisk7
```

Create a GPFS filesystem (cont)

Belisama

- mmcrfs creates a filesystem
  - Need to specify mountpoint, devicename and disks
  - Other options.............

```
# mmcrfs /gpfs gpfs0 -F /tmp/diskdesc.txt -M2 -R 2
```

```
-s roundRobin           Stripe method
-f 8192                 Minimum fragment size in bytes
-i 512                  Inode size in bytes
-I 16384                Indirect block size in bytes
-m 1                    Default number of metadata replicas
-M 1                    Maximum number of metadata replicas
-r 1                    Default number of data replicas
-R 1                    Maximum number of data replicas
-j cluster              Block allocation type
-D posix                File locking semantics in effect
-k posix                ACL semantics in effect
-a 1048576              Estimated average file size
-n 32                   Estimated number of nodes that will mount file system
-B 262144               Block size
-Q none                 Quotas enforced
-F 36864                Maximum number of inodes
-V 8.01                 File system version. Highest supported version: 8.02
-u yes                  Support for large LUNs?
-z no                   Is DMAPI enabled?
-E yes                  Exact mtime mount option
-S no                   Suppress atime mount option
-d hdisk5;hdisk6;hdisk7;hdisk8    Disks in file system
-A no                   Automatic mount option
-o none                 Additional mount options
-T /gpfs1               Default mount point
```

Create a GPFS filesystem (cont)

Belisama

- mmcrfs creates a filesystem
  - Need to specify mountpoint, devicename and disks
  - Other options.............

```
# mmcrfs /gpfs gpfs0 -F /tmp/diskdesc.txt -M2 -R 2

# mmlsfs gpfs0
flag value                description
---- ---------------- -----------------------------------------------------
 -f  4096             Minimum fragment size in bytes
 -i  512              Inode size in bytes
 -I  8192             Indirect block size in bytes
 -m  1                Default number of metadata replicas
 -M  2                Maximum number of metadata replicas
 -r  1                Default number of data replicas
 -R  2                Maximum number of data replicas
 -j  cluster          Block allocation type
 -D  nfs4             File locking semantics in effect
 -k  nfs4             ACL semantics in effect
 -a  1048576          Estimated average file size
 -n  32               Estimated number of nodes that will mount file system
```

Create a GPFS filesystem (cont)

Belisama

- Options (cont)

```
-B  131072                Block size
-Q  user;group;fileset Quotas enforced
    none                  Default quotas enabled
-F  33536                 Maximum number of inodes
-V  10.01 (3.2.1.5)       File system version
-u  yes                   Support for large LUNs?
-z  no                    Is DMAPI enabled?
-L  4194304               Logfile size
-E  yes                   Exact mtime mount option
-S  no                    Suppress atime mount option
-K  whenpossible          Strict replica allocation option
-P  system;goldish       Disk storage pools in file system
-d  sofsnsd1;sofsnsd2;sofsnsd3  Disks in file system
-A  yes                   Automatic mount option
-o  none                  Additional mount options
-T  /gpfs                 Default mount point
```

Create a GPFS filesystem (cont)

Belisama

- Now we just need to mount the filesystem
  - mmmount -a
  - mmmount -N modelist

```
>> NODE: 172.16.1.11 <<
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              6940484   2819492   3762748  43% /
tmpfs                   377872         0    377872   0% /dev/shm
/dev/gpfs0            10485760     49664  10436096   1% /gpfs

>> NODE: 172.16.1.12 <<
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              6940484   2553752   4028488  39% /
tmpfs                   248984         0    248984   0% /dev/shm
/dev/gpfs0            10485760     49664  10436096   1% /gpfs

>> NODE: 172.16.1.13 <<
Filesystem           1K-blocks      Used Available Use% Mounted on
/dev/hda1              6940484   2554320   4027920  39% /
tmpfs                   248984         0    248984   0% /dev/shm
/dev/gpfs0            10485760     49664  10436096   1% /gpfs
```

Disk attributes

Belisama

- mmlsdisk
  - Shows details of the disks that make up a filesystem, failure group, type, status and storage pool (next section)

```
# mmlsdisk gpfs0
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ---------
nsd01        nsd         512      -1 yes      no    ready         up           system
nsd02        nsd         512      -1 no       yes   ready         up           system
Nsd03        nsd         512      -1 no       yes   ready         up           goldish
Attention: Due to an earlier configuration change the file system
is no longer properly replicated.
```

GPFS features and functions - policies

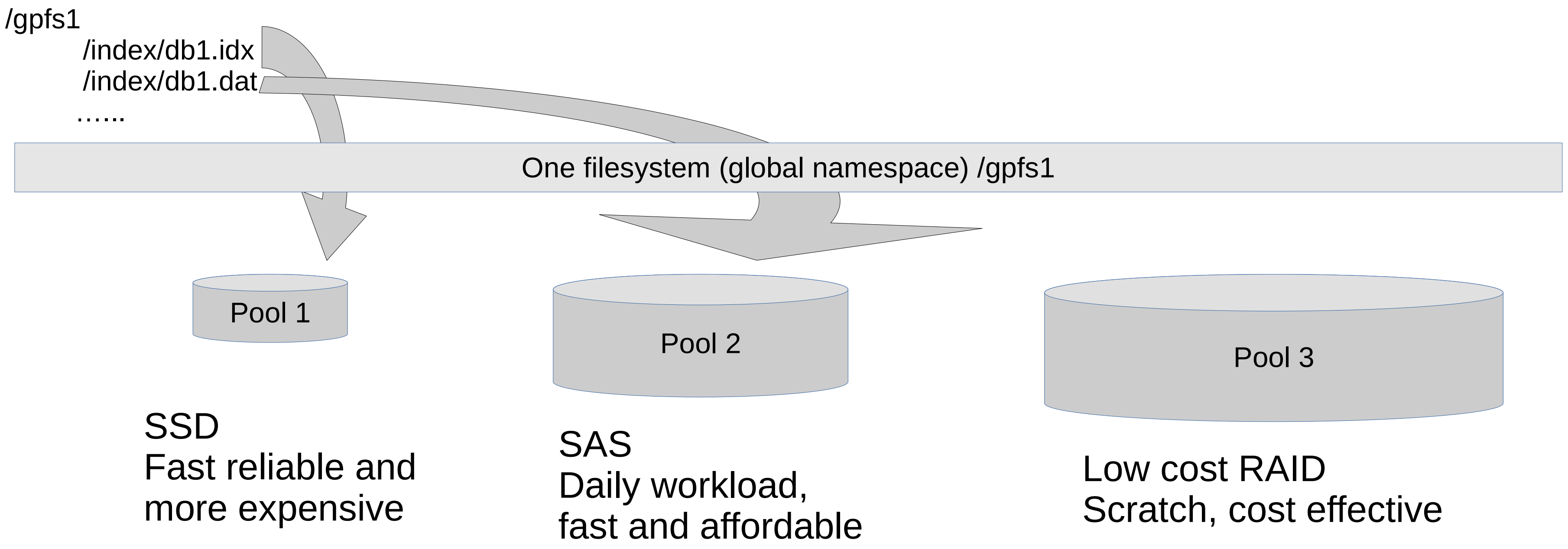
Belisama

- How GPFS handles storage
  - Storage Pools
  - Filesets
  - Policies and placement

Storage pools

Belisama

- A collection of disks or LUNs with similar properties
- Managed together as a group.
- Provide a means to partition the file system's storage


/gpfs1
/index/db1.idx
/index/db1.dat
……
One filesystem (global namespace) /gpfs1
Pool 1
Pool 2
Pool 3
SSD
Fast reliable and more expensive
SAS
Daily workload, fast and affordable
Low cost RAID
Scratch, cost effective

Storage pools (cont)

Belisama

- Motivation
  - Improved price-performance
    - matching the cost of storage to the value of data
  - Improved performance
    - Reducing the contention for premium storage
    - Reducing the impact of slower devices
    - Matching logical block size to physical device characteristics
  - Improved reliability
    - Replication based on need
    - Better failure containment Files
- Maximum of 8 storage pools per Filesystem
- Each disk has this attribute in its disk decriptor
  - At creation time
  - At the time the disk is added to the filesystem
- Files are assigned to storage
  - At creation time
  - Attributes of the file, match the rules of an active policy

Using storage pools

Belisama

- Listing
  - Listing storage pools in a file system uses mmlsfs with –P flag
  - Listing of a file belonging to a pool: mmlsattr

```
# mmlsfs gpfs0 -P
flag value            description
---- ---------------- ------------------------------------------------------
 -P  system;goldish  Disk storage pools in file system
```

```
# mmlsattr new
  replication factors
metadata(max) data(max) file    [flags]
------------- --------- ---------------
      1 (  2)   1 (  2) new
```

```
# mmlsattr -L  new
file name:            new
metadata replication: 1 max 2
data replication:     1 max 2
flags:
storage pool name:    system
fileset name:         root
snapshot name:
```

Using storage pools (cont)

Belisama

- Listing (cont)
  - mmdf –P shows disk utilisation related to pools

```
# mmdf gpfs0  -P system
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 24 GB)
sofsnsd1              5242880       -1 yes      no          5084544 ( 97%)           624 ( 0%)
sofsnsd2              5242880       -1 no       yes         5195392 ( 99%)          1232 ( 0%)
                -------------                         -------------------- -------------------
(pool total)         10485760                              10279936 ( 98%)          1856 ( 0%)
```

Using storage pools (cont)

Belisama

```
# mmdf gpfs0
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 24 GB)
sofsnsd1              5242880       -1 yes      no          5084544 ( 97%)           624 ( 0%)
sofsnsd2              5242880       -1 no       yes         5195392 ( 99%)          1232 ( 0%)
                -------------                         -------------------- -------------------
(pool total)         10485760                              10279936 ( 98%)          1856 ( 0%)

Disks in storage pool: goldish (Maximum disk size allowed is 12 GB)
sofsnsd3              5242880       -1 no       yes         5240704 (100%)           124 ( 0%)
                -------------                         -------------------- -------------------
(pool total)          5242880                               5240704 (100%)           124 ( 0%)

                =============                         ==================== ===================
(data)               10485760                              10436096 (100%)          1356 ( 0%)
(metadata)            5242880                               5084544 ( 97%)           624 ( 0%)
                =============                         ==================== ===================
(total)              15728640                              15520640 ( 99%)          1980 ( 0%)

Inode Information
-----------------
Number of used inodes:            31994
Number of free inodes:             1542
Number of allocated inodes:       33536
Maximum number of inodes:         33536
```

Using storage pools (cont)

- Administration
  - Once a disk is assigned to a storage pool, the pool assignment cannot be changed
  - A root user can change a file’s assigned storage pool by issuing the mmchattr -P command.
    - default is to migrate the data immediately, (can use *-I defer*)
  - The system storage pool can not be deleted
  - A user storage pool is deleted once the last disk is removed from the pool
  - Replicas for storage pools have the same requirements as for filesystems (i.e. failure groups)

Filesets

- Additional object in the Filesystem name space
  - Can be mounted at different points in the GPFS namespace
- Group of objects with independent features
- Can be used to define quotas (data blocks, inodes)
- 10 000 filesets per Filesystem (3.4)
- Define data block and inode quotas at the fileset level
- Apply policy rules to specific filesets
- GPFS 3.5 introduced dependent and independent filesets
  - Dependent
    - Use inodes / space / snapshots from ‘global’ filesystem
  - Independent
    - Own inodes and snapshots, space from ‘global’ filesystem
- root fileset always exists for each filesystem
- mmlsattr –L shows membership
- Newly created files belong to the files of the parent filesystem

Filesets (cont)

- New created fileset
  - Empty directory of the root of the fileset
  - First visible when attached to files
    - mmlinkfileset
    - Attached using „junction“ (similar to hardlinks)
    - One junction per fileset
    - Junction apperas as a directory
- Unlink fileset using mmunlinkfileset
  - Makes files inaccessible
  - Filesets linked below are inaccessible
- Restricted to a single, connected subtree
- Only one root directory
  - No other entries such as hardlinks
  - mv and ln cannot cross fileset boundaries
- Symbolic links can be used
- Filesets and storage pools not specially related
- Quotas: New –j option for GPFS quota commands
  - mmdefedquota, mmdefedquotaon, mmdefedquotaoff, mmedquota, mmlsquota, mmquotaoff, mmrepquota

Filesets (cont)

- Administration
  - – Creating: mmcrfileset
    - Character string < 256
    - Unique within a filesystem
    - root is reserved
  - – Linking: *mmlinkfileset* (creates the junction)
    - Linked to directory
    - Linked to other fileset
  - – Unlinking: *mmunlinkfileset*
  - – Changing: Unlinking and linking with a new junction
  - – Displaying: *mmlsfileset*
    - Shows name, fileset identifier, junction, status, root inode

- Fileset commands:
  - – mmchfileset
    - Change fileset data
  - – mmlsfileset
    - List filesets and information
  - – mmunlinkfileset
    - Removes a association between a junction and a fileset
  - – mmcrfileset
    - Creates a fileset definition
  - – mmdelfileset
    - Deletes a filest definition
  - – mmlinkfileset
    - Assings filesets a junction

Policies

Belisama

- Policy: Set of rules defining the life cycle of user defined data
- Automate management of files using policies and rules
  - File placement policies: Where to place newly created files
  - File management policies: When to move or delete files
- Automate management of files using policies and rules
- Placement policies
  - Rules within a policy file
    - One active placement policy at a time
    - Can contain any number of rules
    - Not larger than 1 MB
  - First creation of GPFS Filesystem: System storage pool
- File management policies
  - Migration and deletion: mmapplypolicy
    - Using a separate management policy file

Policies (cont)

- Example file management rule

  -
    ```
    RULE [’rule_name’] SET POOL ’pool_name’
    [ REPLICATE(data-replication) ]
    [ FOR FILESET( ’fileset_name1’,
    ’fileset_name2’, ... )]
    [ WHERE SQL_expression ]
    ```
  - Where
    - RULE: Initiates with optional rule_name
    - SET POOL: Name of the pool to place data on
    - REPLICATE: Override replication settings (0,1)
    - FOR FILESET: Optional for specific filesets
    - WHERE: SQL Expression
      - acess_age, file_size, day_of_month, access_time

Policies (cont)

```
define(stub_size,0)
define(is_premigrated,(MISC_ATTRIBUTES LIKE '%M%' AND KB_ALLOCATED > stub_size))
define(is_migrated,(MISC_ATTRIBUTES LIKE '%M%' AND KB_ALLOCATED == stub_size))
define(access_age,(DAYS(CURRENT_TIMESTAMP) - DAYS(ACCESS_TIME)))
define(mb_allocated,(INTEGER(KB_ALLOCATED / 1024)))
define(exclude_list,(PATH_NAME LIKE '%/.SpaceMan/%' OR
                     NAME LIKE '%dsmerror.log%' OR PATH_NAME LIKE '%/.ctdb/%'))
define(weight_expn,(CASE WHEN access_age < 1 THEN 0
                         WHEN mb_allocated < 1 THEN access_age
                         WHEN is_premigrated   THEN mb_allocated * access_age * 10
                         END))

RULE defaultmig MIGRATE FROM POOL 'system' THRESHOLD (80,75)
WEIGHT(weight_expn) TO POOL 'hsm' WHERE NOT (exclude_list) AND
NOT (is_migrated)
```

Commands

- Policy commands:
  - mmapplypolicy
    - Applies and tests policies
  - mmchpolicy
    - Create and test policies for a filesystem
  - mmlspolicy
    - List policies and information
  - mmrestripefile
    - Re-balance files within storage pools or for storage pools

GPFS features and functions - availability

- Quorum
- Filesystem considerations
- Failure Groups
- Replication
- HA and DR

HA and DR

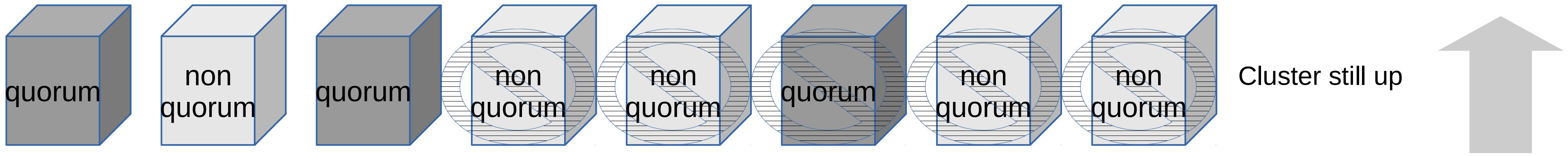
Belisama

- Two servers can not decide between “the other server is down” and “the communication to the other server is down”
  - An independent decision maker is required, e.g.
    - Manual operator intervention
    - Tie breaker
    - Quorum concept: (n/2)+1
- Same is true for mirrored disks: A quorum of disks guarantees the integrity of filesystem metadata
- High Availability and Disaster Resilience
  - It‘s NOT that easy and simple
  - The devil is in the detail

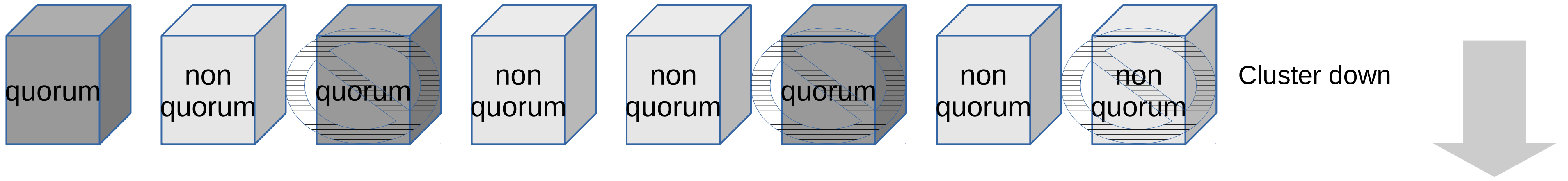
Quorum

- GPFS quorum = ½ (number of quorum nodes) + 1
- Available since GPFS version 2.2
- Usually the most reliable nodes
- odd number recommended

Quorum = 3, Failed nodes = 5



Quorum = 3, Failed nodes = 3

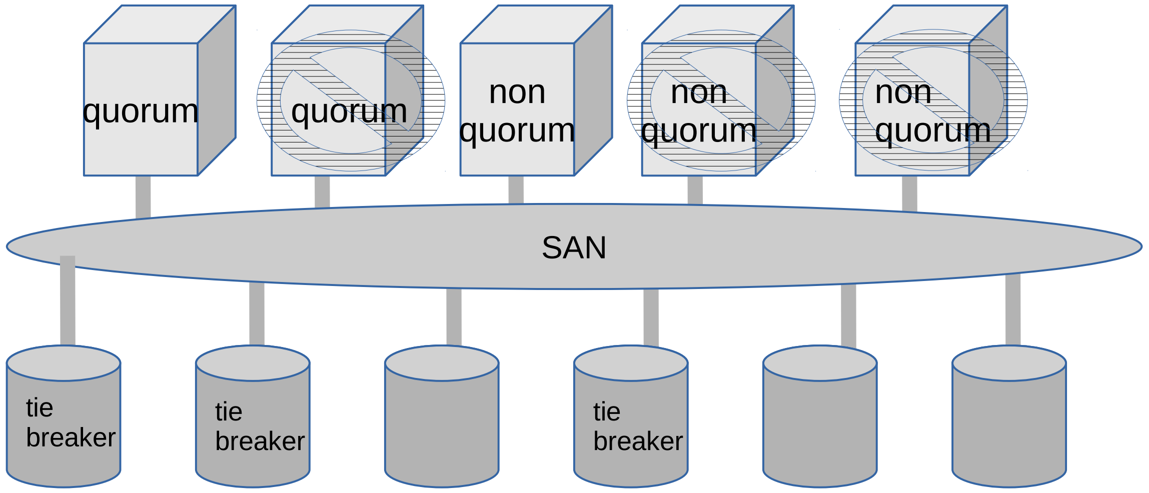


Quorum (cont)

- Managing quorum nodes
  - At creation time either specify quroum in node file or at command line:
    - *mmcrcluster –N myhostname:**quorum**:myadminlan*
  - During operations:
    - *mmchconfig designation=quorum –N nodename*
  - A quorum node changed to a non-quorum node must have GPFS stopped on it
  - Showing current quorum nodes:
    - *mmlscluster*

Quorum (cont)

- Tiebreaker disks
  - The concept of tiebreaker disks added in GPFS version 2.3 for small clusters
  - 1 to 3 tiebreaker disks directly attached to the core quorum nodes



Cluster still up

Quorum (cont)

- Managing quorum with tiebreaker disks
  - Same procedure as before except
  - Only two to eight quorum server are allowed
  - Tiebreaker disks need to switched on using

    mmchconfig tiebreakerDisks=“nsd1;nsd2;nsd3“
    - 1 -3 disks allowed
    - NSD disks
    - Separator is ;
    - Enclosed in double quotes
  - Tiebreaker disks can be switched off using

    mmchconfig tiebreakerDisk=no

Filesystem quorum

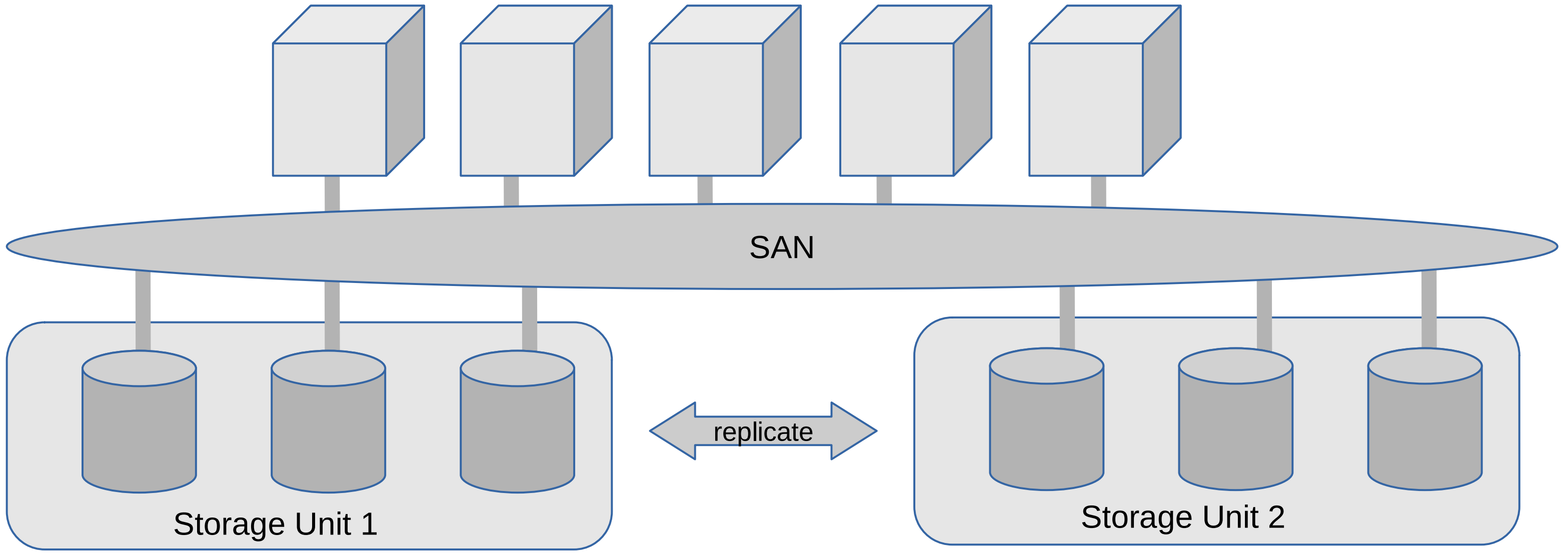
Belisama

- Node quorum determines if the cluster will remain active
- Filesystem quorum determines if the filesystem will remain mounted
  - There is a structure in GPFS called the file system descriptor that is initially written to every disk (NSD) in the file system, but is replicated on a subset of the disks as changes to the file system occur, such as adding or deleting disks. Based on the number of failure groups and disks, GPFS creates between one and five replicas of the descriptor:
    - If there are at least five different failure groups, five replicas are created.
    - If there are at least three different disks, three replicas are created.
    - If there are only one or two disks, a replica is created on each disk.
    - Once it is decided how many replicas to create, GPFS picks disks to hold the replicas, so that all replicas will be in different failure groups, if possible, to reduce the risk of multiple failures. In picking replica locations, the current state of the disks is taken into account. Stopped or suspended disks are avoided.
    - Similarly, when a failed disk is brought back online, GPFS may modify the subset to rebalance the file system descriptors across the failure groups. The subset can be found by issuing the mmlsdisk -L command.
    - GPFS requires a majority of the replicas on the subset of disks to remain available to sustain file system operations:
      - If there are at least five different failure groups, GPFS will be able to tolerate a loss of two of the five groups. If disks out of three different failure groups are lost, the file system descriptor may become inaccessible due to the loss of the majority of the replicas.
      - If there are at least three different failure groups, GPFS will be able to tolerate a loss of one of the three groups. If disks out of two different failure groups are lost, the file system descriptor may become inaccessible due to the loss of the majority of the replicas.
      - if there are fewer than three failure groups, a loss of one failure group may make the descriptor inaccessible.
      - If the subset consists of three disks and there are only two failure groups, one failure group must have two disks and the other failure group has one. In a scenario that causes one entire failure group to disappear all at once, if the half of the disks that are unavailable contain the single disk that is part of the subset, everything stays up. The file system descriptor is moved to a new subset by updating the remaining two copies and writing the update to a new disk added to the subset. But if the downed failure group contains a majority of the subset, the file system descriptor cannot be updated and the file system will be force unmounted.

Failure groups

Belisama

- disks having the same single point of failure should be assigned to the same failure group during configuration



Failure groups (cont)

- Can be defined at creation time
  - *mmcrnsd –F diskdesc.txt*
- Can be changed using mmchdisk
  - *mmchdisk gpfs1nsd change –d “::::455:::“*
- *Changing failure groups example*
  - Changing disks to be in a new failure group
    - Can be changed for existing disks
    - Can be defined at creation time

```
%nsd:
nsd=NsdName
usage={dataOnly | metadataOnly | dataAndMetadata | descOnly}
failureGroup=FailureGroup
pool=StoragePool
servers=ServerList (NSDSrv1,NSDSrv2,..(8))
device=DiskName
```

Failure groups (cont)

Belisama

- Example of system with two failure groups

```
# mmlsdisk gpfs0
disk         driver   sector failure holds    holds                             storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
nsd01        nsd         512       1  yes      yes   ready         up           system
nsd02        nsd         512       2  yes      yes   ready         up           system
```

Replication

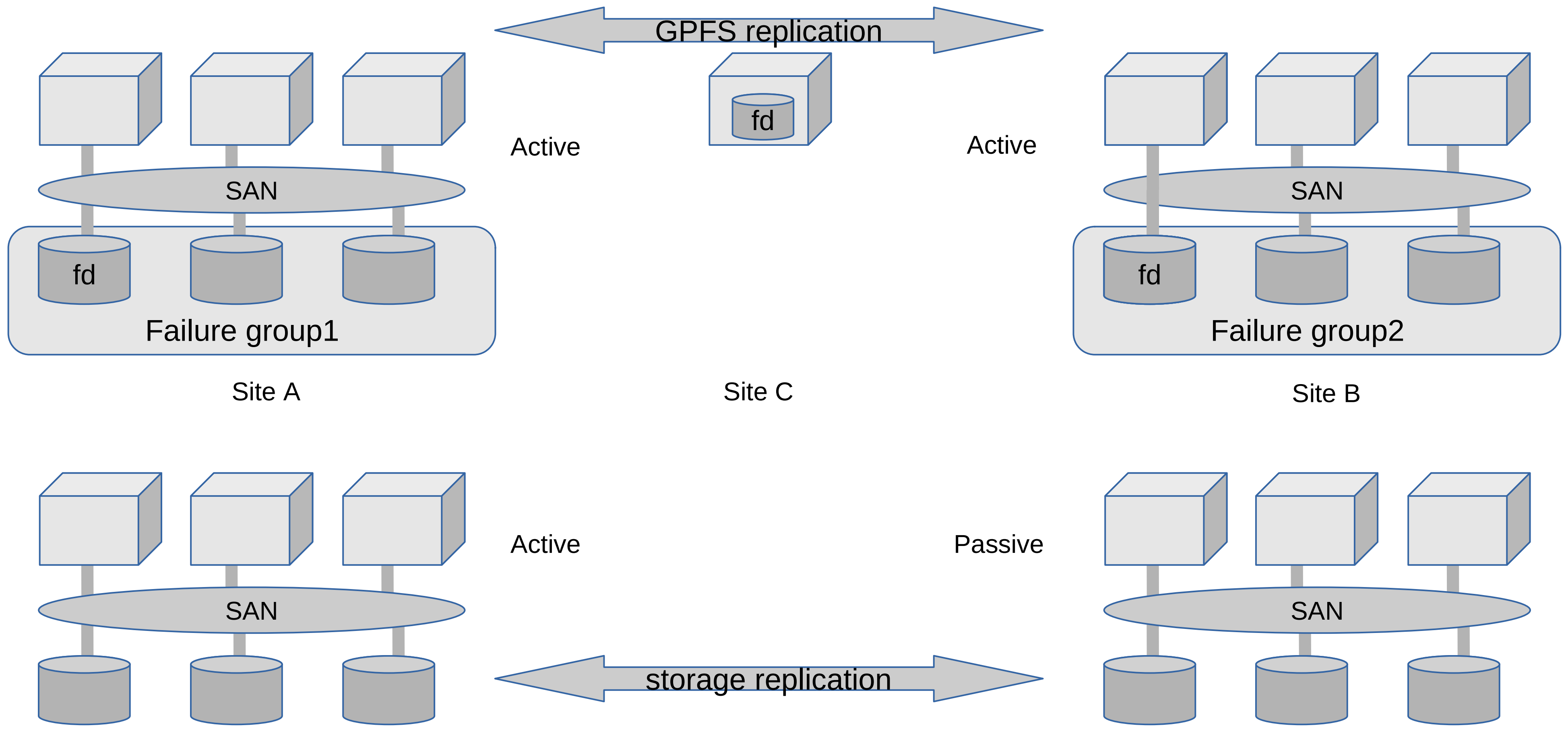
Belisama

- Replication can be specified at creation time
  - *mmcrfs /gpfs2 gpfs2 -F /tmp/gpfs2dsk -n 24 -m 2 -M 2 -r 2 -R 2*
- To change it later, however the maximum values –M and –R must be set to 2 or 3 when the filesystem was created and cannot be changed
  - *mmchfs gpfs2 –r 2 –m 2*
- It is possible to replicate only certain files
  - *mmchattr -m 2 -r 2 /fs1/project7.resource*
- Verify replication using mmlsfs or mmlsattr

```
# mmlsfs gpfs0
flag value                description
---- ---------------- -----------------------------------------------------
....
 -m  1                Default number of metadata replicas
 -M  2                Maximum number of metadata replicas
 -r  1                Default number of data replicas
 -R  2                Maximum number of data replicas
...
```

DR options

Belisama



GPFS monitoring

Belisama

- Look at:
  - Monitor global system health
  - Display disk usage
  - Monitor GPFS performance

GPFS monitoring

Belisama

- Global health

```
 # mmgetstate -Lsa

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state  Remarks
-------------------------------------------------------------------------------------
       1      ts1                2        3          3        active      quorum node
       2      ts2                2        3          3        active      quorum node
       3      ts3                2        3          3        active      quorum node

 Summary information
---------------------
Number of nodes defined in the cluster:            3
Number of local nodes active in the cluster:       3
Number of remote nodes joined in this cluster:     0
Number of quorum nodes defined in the cluster:     3
Number of quorum nodes active in the cluster:      3
Quorum = 2, Quorum achieved
```

- Disk usage

```
# mmdf gpfs0 -P goldish
disk                disk size  failure holds    holds              free KB             free KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: goldish (Maximum disk size allowed is 12 GB)
sofsnsd3              5242880       -1 no       yes          5240704 (100%)           124 ( 0%)
                -------------                         -------------------- -------------------
(pool total)          5242880                                5240704 (100%)           124 ( 0%)
```

mmpmon

Belisama

email me for example scripts.

- Different modes
  - Up to 5 instances of mmpmon allowed
  - I/O mode
    - io_s: Shows total IOs
    - fs_io_s: Shows IOs for a filesystem
    - Filesystem level or node level, output includes:
    - Number of disks; time stamp; bytes read and written; file open, close, read, write; readdir and inode updates

```
ver
fs_io_s
```

```
mmpmon node 172.2.1.23 name s7801p23  fs_io_s OK
cluster: asguard
filesystem: gpfs1
disks: 4
timestamp: 1121974088/463102
bytes read: 24559
bytes written: 8748
opens: 289
closes: 209
reads: 2668
writes: 146
readdir: 29
inode updates: 22
.....
```

```
_fs_io_s_ _n_ 172.16.1.11 _nn_ ts1 _rc_ 0 _t_ 1248761263
 _tu_ 380682  _cl_ sofs151.red.com _fs_ gpfs0 _d_ 3 _br_ 0
_bw_ 0 _oc_ 1029 _cc_ 1029
```

  - Histogram mode
    - Specify the size ranges (in bytes of i/o) and latency in milliseconds
    - Output can be human or “machine” readable.

```
rhist on
rhist nr 512;1m;4m  1;5;10
rhist off
```

```
size range                  0 to    255      count       80625
    latency range         0.0 to      1.0    count        1476
    latency range         1.1 to     10.0    count       28445
    latency range        10.1 to     30.0    count       39775
    latency range        30.1 to    100.0    count       10093
    latency range       100.1 to    200.0    count         834
    latency range       200.1 to        0    count           2
 size range               256  to    1023    count     2398875
...
```

Administration

Belisama

- Reminder
  - Design test clusters (training? dev?)
  - Change control – plan outage windows years in advance, keep detailed change control records.
  - You are managing a cluster (Definition many instances of the operating system that appear to the end user as the same system).
- GPFS GUI
  - Cluster and Storage administration
  - Monitoring (pmcollectors and pmsensors)
  - Easy to integrate with Grafana / time series database
- Nigel's tool njmon is has started to collect GPFS metrics as well

GPFS tools

Belisama

- The following tools are provided by IBM (found in /usr/lpp/mmfs/samples)
- Network performance testing
  - nsdperf: A simple tool to test network performance under load (no disk access) – better at handling multiple nodes than iperf.
- I/O performance
  - gpfsperf: A simple tool to measure GPFS performance using several common file access patterns

GPFS features and functions (cont)

- Snapshots
- Data Management API (DMAPI)
- GPFS and hierarchical storage management (HSM)
- Remote mount capabilities

GPFS features and functions (cont)

Belisama

- Creating a snapshot

  # mmcrsnapshot fs1 snap1

  writing dirty data to disk....

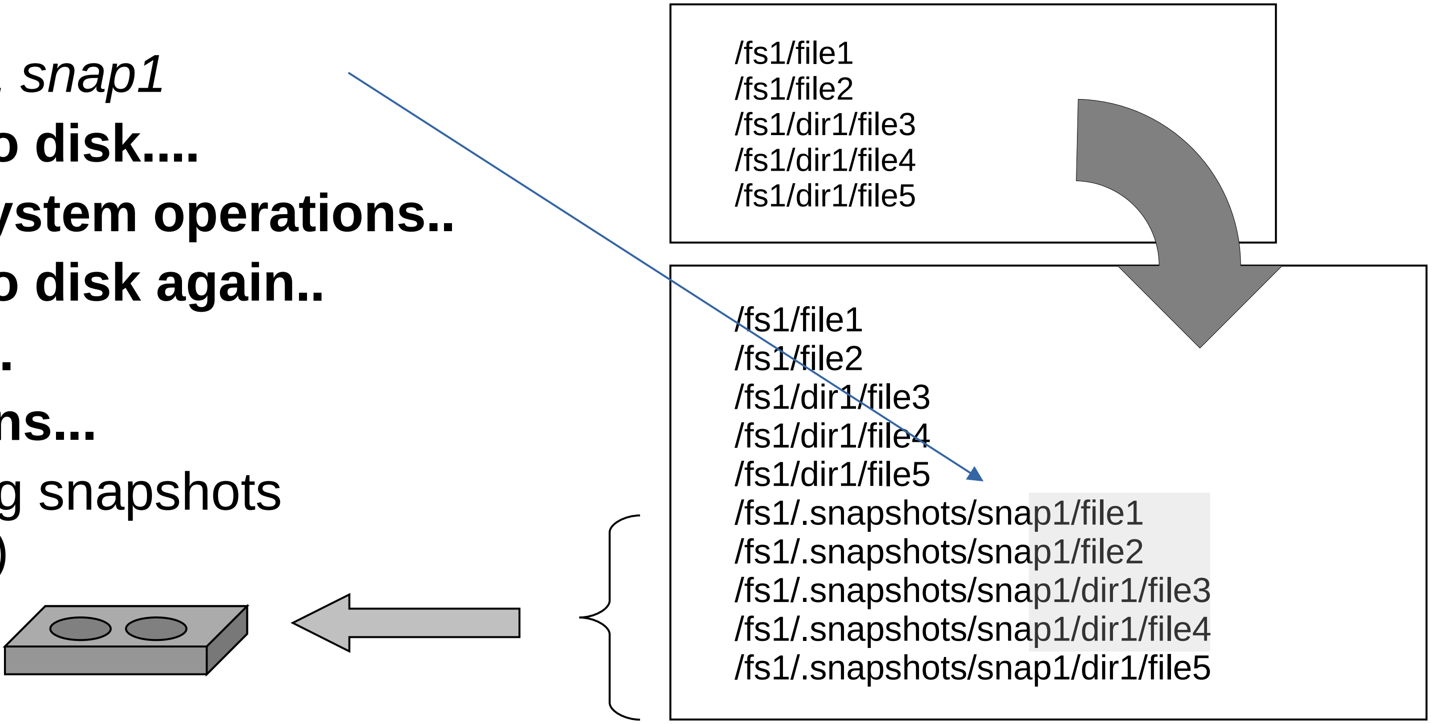
  quiescing all file system operations..

  writing dirty data to disk again..

  creating snapshot..

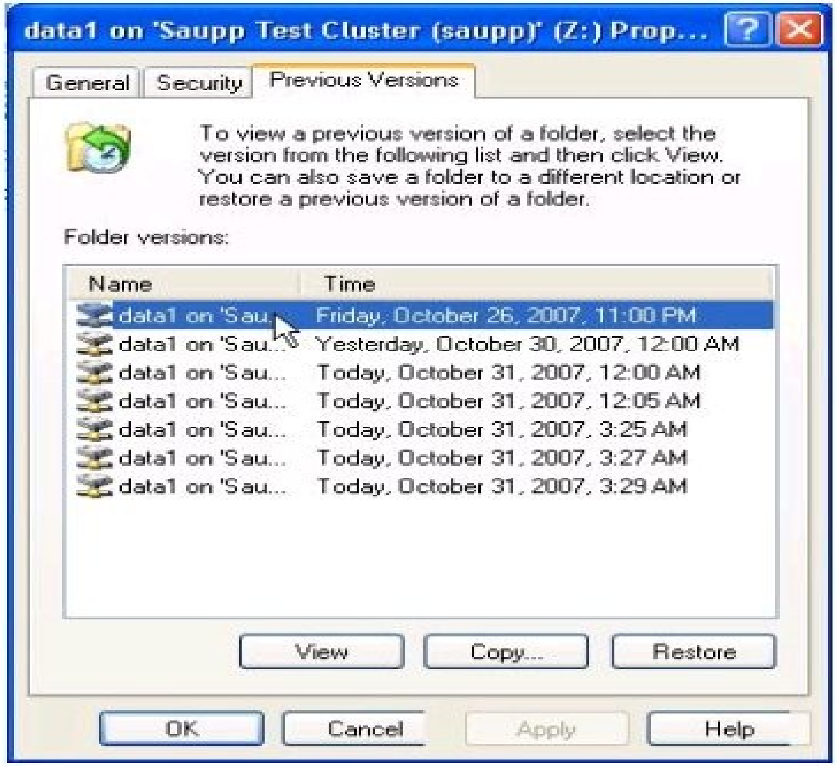
  resuming operations...

- Up to 256 outstanding snapshots (performance impact)


/fs1/file1
/fs1/file2
/fs1/dir1/file3
/fs1/dir1/file4
/fs1/dir1/file5
/fs1/file1
/fs1/file2
/fs1/dir1/file3
/fs1/dir1/file4
/fs1/dir1/file5
/fs1/.snapshots/snap1/file1
/fs1/.snapshots/snap1/file2
/fs1/.snapshots/snap1/dir1/file3
/fs1/.snapshots/snap1/dir1/file4
/fs1/.snapshots/snap1/dir1/file5

Read only copy, only changes to original file use disk space

GPFS features and functions (cont)


data1 on 'Saupp Test Cluster (saupp)' (Z:) Prop...

General | Security | Previous Versions

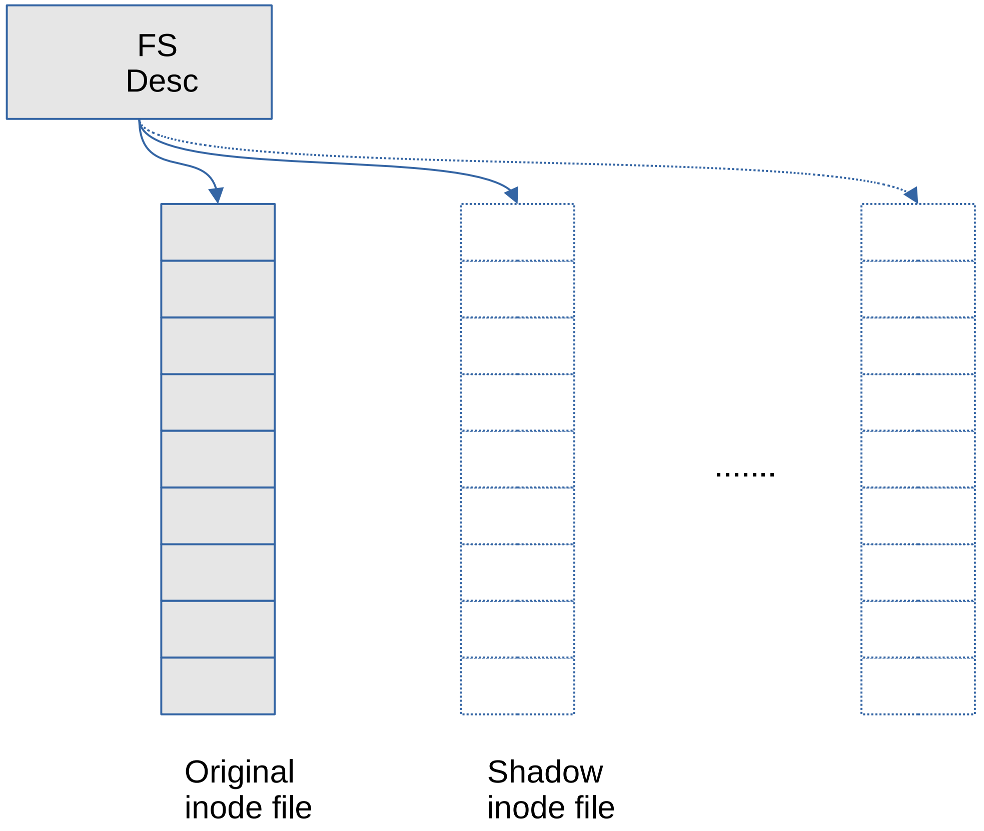
To view a previous version of a folder, select the version from the following list and then click View. You can also save a folder to a different location or restore a previous version of a folder.

Folder versions:

Name	Time
data1 on 'Sau...	Friday, October 26, 2007, 11:00 PM
data1 on 'Sau...	Yesterday, October 30, 2007, 12:00 AM
data1 on 'Sau...	Today, October 31, 2007, 12:00 AM
data1 on 'Sau...	Today, October 31, 2007, 12:05 AM
data1 on 'Sau...	Today, October 31, 2007, 3:25 AM
data1 on 'Sau...	Today, October 31, 2007, 3:27 AM
data1 on 'Sau...	Today, October 31, 2007, 3:29 AM

- Snapshot is integration into windows using Volume Shadow Copy Service

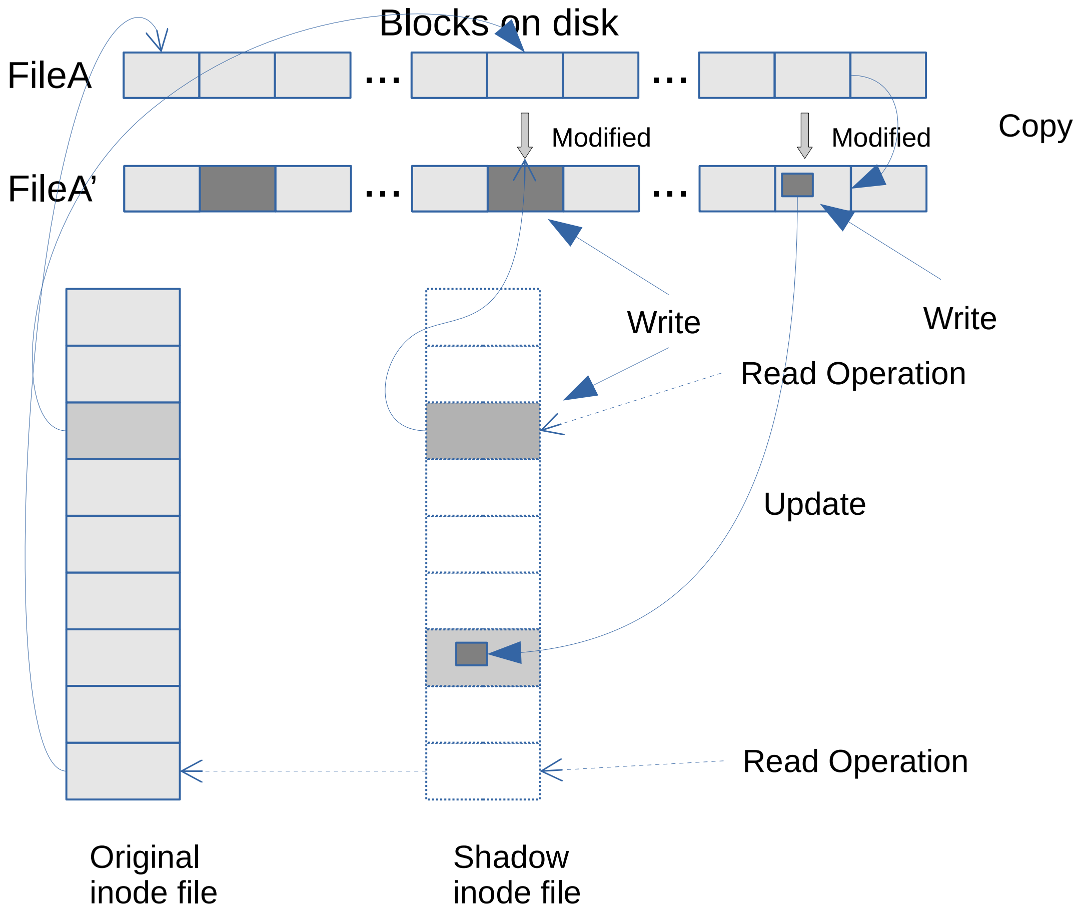
GPFS features and functions (cont)



- Flush dirty data
- Quiesce filesystem operations
- Flush dirty data
- Create sparse shadow inode file
- Add entry to snapshot table in FS descriptor

GPFS features and functions (cont)

Belisama



- Snapshots
  - COWOWYHTOROW*
  - Can restore from them
  - Can snapshot filesets (independant)
  - Integrated with mmbackup (only works with Spectrum Protect)
- Operation
  - For most data operations new snapshot data in GPFS is directed into new data blocks and pointers are changed for the version of the file being modified.
  - In the case where less than a GPFS file system block is modified GPFS creates a new block and copy over the unchanged data.

* copy-on-write-only-when-you-have-to-otherwise-redirect-on-write"

GPFS features and functions (cont)

Belisama

- The Open Group has defined a standard API that allows to create extensions to existing file systems
  - Data Storage Management (XDSM) API Common Applications Environment (CAE) Specification C429
  - GPFS has implemented this standard except some optional features
  - DMAPI for GPFS allows to monitor events associated with a GPFS file system or with an individual file and to manage and maintain file system data
- The GPFS DMAPI in combination with other products provides
  - Hierarchical storage management (in combination with IBM Spectrum Protect for Space Management, also known as HSM)
- Data Management API (DMAPI) support with HSM

– Files can be migrated to tape storage pool, leaving a stub in the filesystem

– When stub is accessed, a recall is issued.

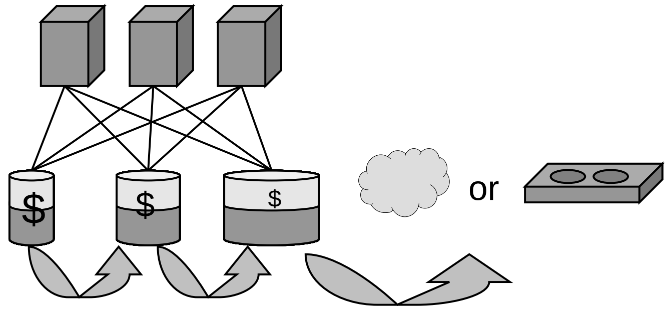
– GPFS does the scanning

  - Example: Rules when file ages, size of file, filesystem fullness etc

- GPFS and hierarchical storage management (HSM)

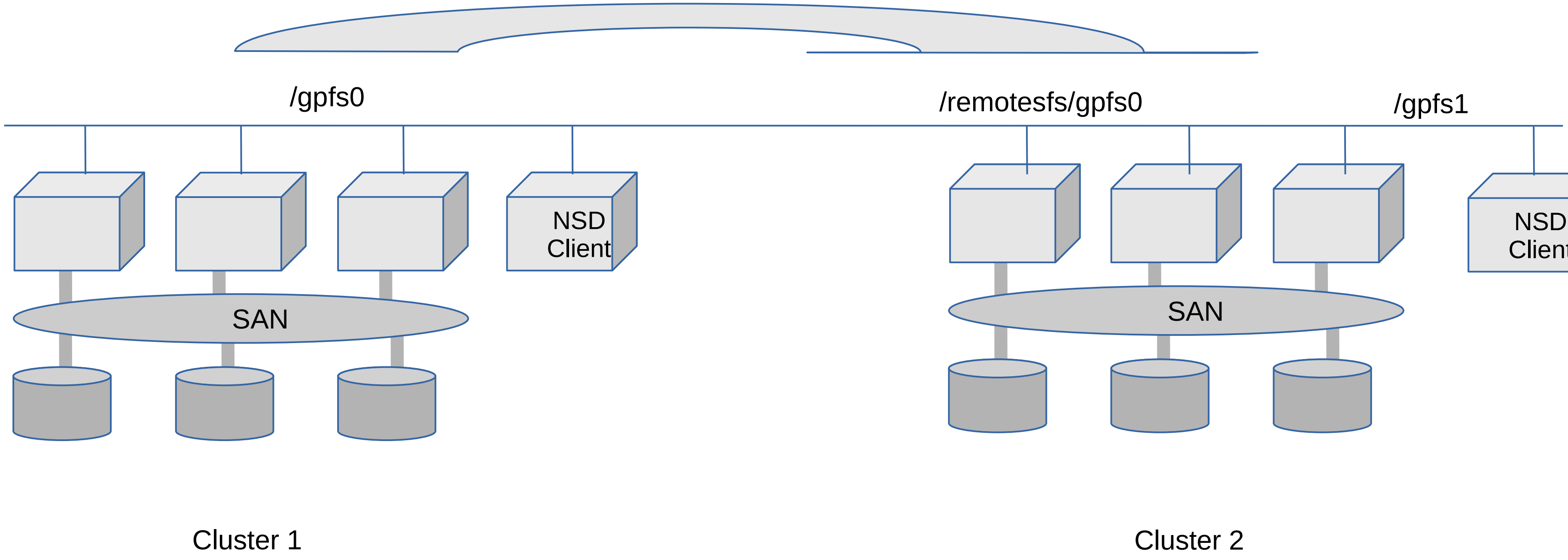
– Leave stub file on disk

– When accessed, initiates a recall from tape / Cloud storage



GPFS features and functions (cont)

- Remote mount capabilities



GPFS Management

- Spectrum Scale is a fully clustered filesystem with CLI / GUI management tools
  - Add and remove nodes
  - Add and remove underlying storage
  - Add and remove file systems, filesets and pools
  - Managing policies
  - Perform rolling upgrade through the cluster
- Monitoring and performance metrics
  - Monitoring the general health of the cluster, nodes, filesystems and filesets
  - Monitoring the health of the hardware and managing call home

Some changes in version 5.x

Belisama

- Core improvements
  - worked on performance acceleration via RDMA, enhanced metadata performance, improved the handling of small files (stat cache) and small file space efficiency.
  - Compression has been optimised with support for LZ4 compression (and offload for power). By file, pool, fileset etc
  - On Linux startup will detect changes in Kernel and rebuild
  - Improvements in LTFS EE integration like optimisation in order to reduce the possibility of recall storms during backups
  - *mmnetverify* now supports remote clusters.
  - Now Spectrum Scale uses a lenient round-robin algorithm which makes rebalancing much faster vs the strict round-robin method used in earlier versions.
  - While doing a file system integrity check, if the mmfsck command is running for a long period of time, another instance of mmfsck can be launched with the –stats-report option to display current status from all the nodes that are running the mmfsck command.
  - Spectrum Scale cluster health check commands have been enhanced with options to verify file system, SMB and NFS nodes.
  - The mmcallhome command has a new option '–pmr' which can be used to specify an existing PMR number for data upload.
  - Spectrum Scale installation toolkit was introduced with version 4.1 and many enhancements are made in Version 5.0. The installation kit now supports deploying protocol nodes in a cluster that uses Spectrum Scale Elastic Storage Server (ESS). The installation toolkit also supports configuring Call Home and File Audit Logging. Deployment of Ubuntu 16.04 LTS nodes as part of the cluster are also supported by the installation toolkit.
- Security
  - Introducing File Audit Logging logs filesystem events to a retention-enabled fileset to track user access to the file systemFile
  - Audit logging that was introduced in 5.0 release, now has multi-cluster (remote mount) support and support for IBM System Z.
  - Enhanced usability for secure data at rest (encryption)

Some further changes in version 5.x

- Watch Folders
  - An much awaited feature that providers flexible API which allows programmatic actions to be taken based on filesystem events. Can be run against directories, filesets, and inode spaces. For Use of this feature for your use case in production , contact IBM.
- Deployment toolkit
  - Designed to simplify GPFS deployments now has support for System Z, Ubuntu 18.04/18.04.1 and support for file audit logging along with watch folders.
  - Improvements in the toolkit for different upgrade scenarios
- Operating systems
  - Currency for Ubuntu 18.04.1 kernel support
  - Windows 10 Enterprise Edition client support was added.

Some further changes in version 5.x

- Protocols
  - Dynamic modification of NFS exports and support sor NFSv4 pseudo path
  - Improved upgrade support for Object
  - Ubuntu support for protocol nodes (NFS/SMB/Object)
- Management GUI
  - Enhancement to manage/configure AFM & TCT
  - Network monitoring for both IP and RDMA transports
  - Upload diagnostic data to a PMR automatically, etc.
  - includes quota and capacity monitoring of remote clusters, ability to enable/disable File audit logging and security fixes which includes logging off of users on change of passwords or roles, etc.
- REST API
  - Expanded REST API for Performance data collection, threshold management, snap creation, addition/removal of nodes from cluster.
  - Support for change and retrieval of SMB ACL and Configuration/ management of File audit logging.
- Big Data and Analytics
  - Certification with HortonWorks Data Platorm 2.6 (5.0) and in 5.02 there is support for Hortonworks Data Platform 3.0 and Management Pack 2.7.0.0, Support for Apache Hadoop 3.0.x, Support for native HDFS encryption and improvements in FPO based setup for scanning of inconsistent replicas.
- Transparent Cloud Tiering
  - Remote mounted filesystem support, tier different fileset to different cloud containers, enhanced support for multiple cloud accounts and containers.

New features

- AFM
  - Data sharing - GPFS includes active file management, which is a scalable, high-performance remote file data caching solution that is integrated within a GPFS file system. If you have a situation where massive amounts of data is gathered at separate locations and the results are analysed by people at other locations, you need a solution that makes it possible to transparently move file data automatically to where it is needed.
  - This is especially useful for collaborative projects, applications and workflows that are managed globally, but need to have access to the same files.
- GPFS Raid
  - Greater throughput; faster rebuild times; end to end checksum
- GPFS File Placement Optimisation
  - GPFS Shared nothing clusters
- High Performance Extended Attributes
  - GPFS has long supported the use of extended attributes, though in the past they were not commonly used, in part because of performance concerns. In GPFS 3.4, a comprehensive redesign of the extended attributes support infrastructure was implemented, resulting in significant performance improvements. In GPFS 3.5, extended attributes are accessible by the GPFS policy engine allowing you to write rules that utilise your custom file attributes.
  - Now an application can use standard POSIX interfaces to manage extended attributes and the GPFS policy engine can utilise these attributes.

GPFS Active File Management

Belisama

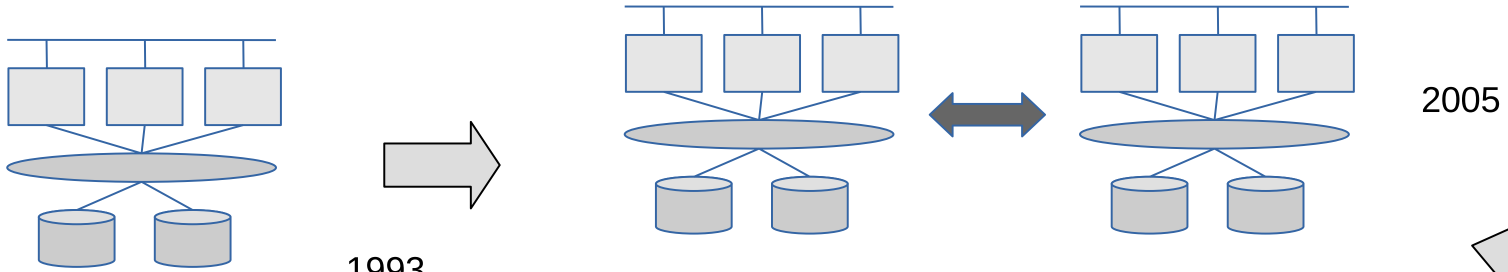
- When GPFS was introduced in 1998 it represented a revolution in file storage. For the first time a group of servers could share high performance access to a common set of data over a SAN or network. The ability to share high performance access to file data across nodes was the introduction of the global namespace.
- Later GPFS introduced the ability to share data across multiple GPFS clusters. This multi-cluster capability enabled data sharing between clusters allowing for better access to file data. This further expanded the reach of the global namespace from within a cluster to across clusters spanning a data centre or a country.
- There were still challenges to building a multi-cluster global namespace. The big challenge is working with unreliable and high latency network connections between the servers. Active File Management(AFM) in GPFS addresses the WAN bandwidth issues and enables GPFS to create a world-wide global namespace. AFM ties the global namespace together asynchronously providing local read and write performance with automated namespace management. It allows you to create associations between GPFS clusters and define the location and flow of file data.

AFM Improvements in 5

- Support for File Compression for AFM and AFM DR filesets
- Load balancing enhancements
- ILM support for snapshots for AFM and AFM DR filesets
- AFM enhancements to modify gateway nodes for a fileset
- AFM pre-fetch option enhancements and read-only AFM relationships using read-only NFS exports

What is Active File Management (was Panache)

Belisama



1993

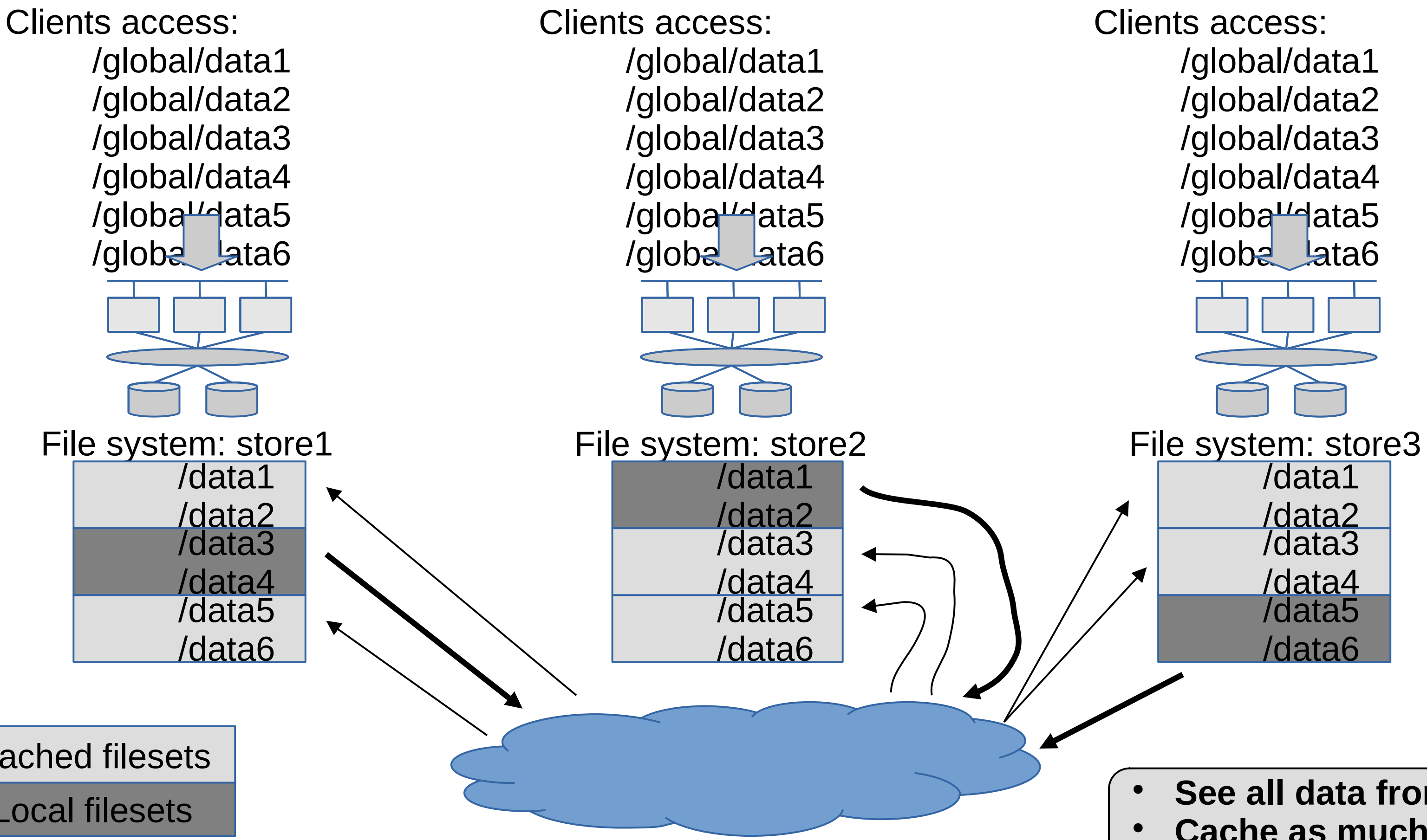
GPFS introduced concurrent file system access from multiple nodes.

Multi-cluster expands the global namespace by connecting multiple sites

- AFM takes global namespace truly global by automatically managing asynchronous replication of data
- If data is in cache …
  - Cache hit at local disk speeds
  - Client sees local GPFS performance
    if file or directory is in cache
- If data not in cache …
  - Data and metadata (files and directories) pulled on-demand at network line speed and written to GPFS
  - Uses NFS/pNFS for WAN data transfer

Global namespace with AFM Cache

Belisama



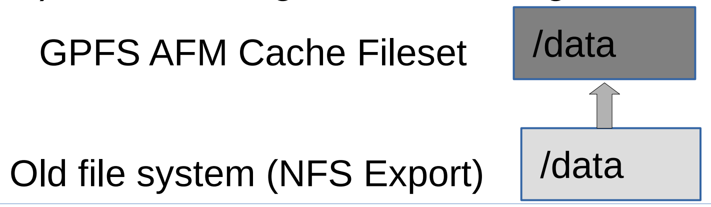
▨ Cached filesets
■ Local filesets

- **See all data from any cluster**
- **Cache as much data as required or fetch data on demand**

Migrate data into GPFS using AFM

Belisama

- You can use AFM to migrate data from an NFS source into a Spectrum Scale file system. There are two options:
  - Local-update (LU)
    - doesn't push changes back to the home file system, therefore no easy roll back
    - Using local-update data is read from the Home (original) and copied into the cache fileset on demand or using prefetch. You can move active users over before all of the data is prefetched but you need to prefetch the metadata before cutting over completely.
  - Independant-writer (IW)
    - Keeps data in the original file system up to date (therefore additional IO)
    - Migrating data using Independent-writer (IW) mode makes sense if you have the bandwidth to push changes (after application cut-over) to Home. Using independent-writer data is read from the Home (target) and copied into the cache fileset on demand or using prefetch. You can move active users over before all of the data is prefetched . With IW mode you do not have to prefetch the metadata before cutting over the application and you can fail-back at any time. If you do fail-back to the Home I suggest issuing a flushpending (and waiting for it to complete) before failing-back so changes made in the cache are not missed at the Home.



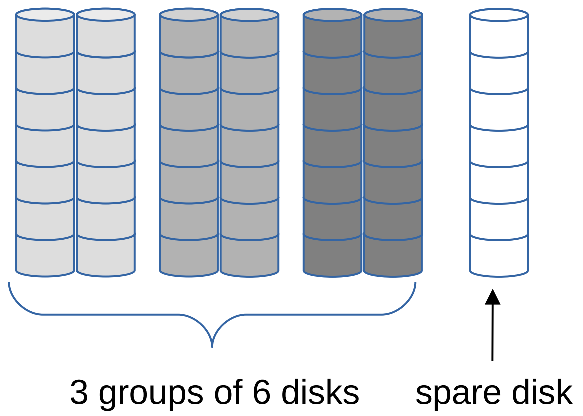
Why GPFS Native Raid

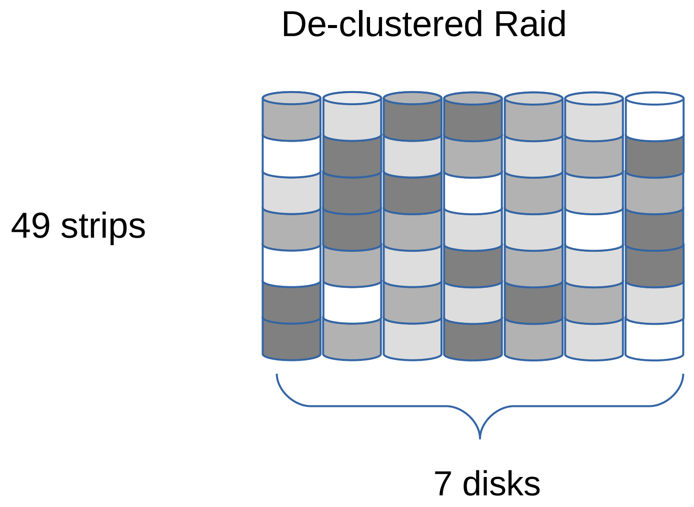
- When building a cluster with multiple pods – slow rebuild of one pod, will affect performance of not only that pod but whole filesystem (as reads / writes spread across all pods)
- Disks bigger, takes longer to rebuild
- With such a large number of disks, likelihood of failure greater
- Silent data corruption also more common
- What we achieve:
  - Stack from application / gpfs / raid controller / disk to application / gpfs+raid controller / disk
  - De-clustered array removes rebuild, but also has end to end checksum to protect against data corruption. Silent (phantom) errors – Not media errors, these the disk can tell you that there is an error, for silent errors, it doesn't know.
    - Far or near off-track writes (vibration / thermal, head misses), dropped writes, Head doesn't check at time of writing. Only see affect at time of reading. Also have undetected read errors.
  - Almaden estimate 1000 disk system will experience 1 error every 5 years
    - Read block – gives A – good, but if read B – problem (no data better than bad data)
  - We now attach a checksum to data, so we can check data, but this will not check dropped writes (as old data will match the checksum). To protect against this, we put the checksum at a different location (use version number and checksum)
  - We are a RAID controller – so need to as well as rebuild, rebalance, scrub and control the scheduling of these operations (setting rate on criticality of the rebuild etc)

De-clustered RAID example

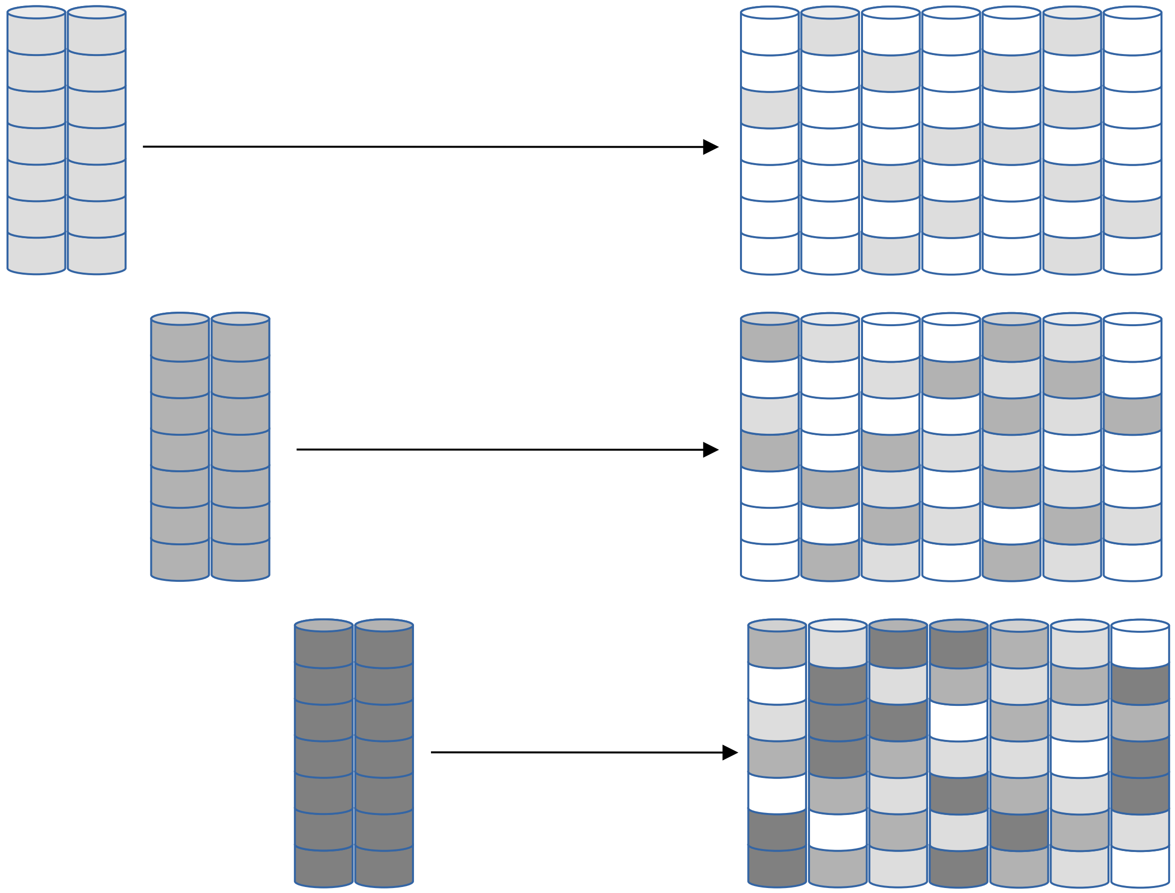
Belisama

- 3 fault tolerant mirrored groups (RAID 1)
  - 7 stripes per group
  - 2 strips per stripe



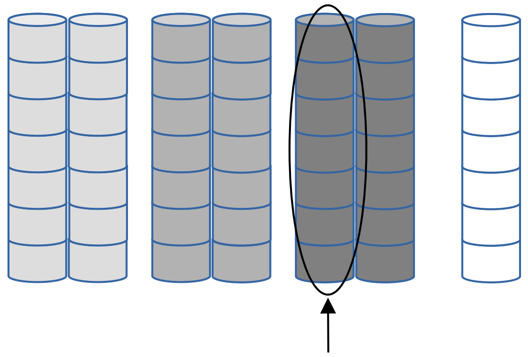


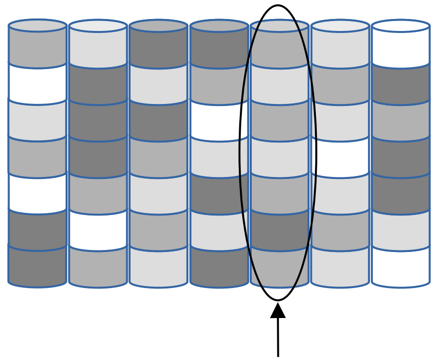
De-clustered RAID Example



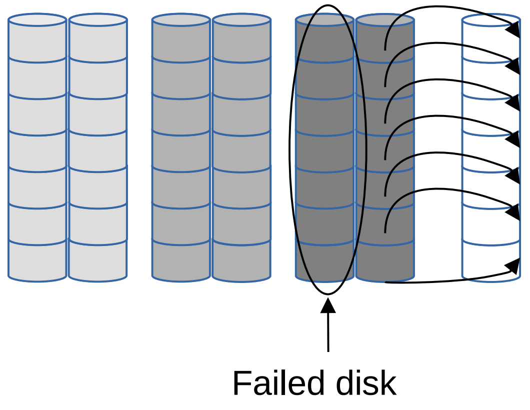
De-clustered RAID example

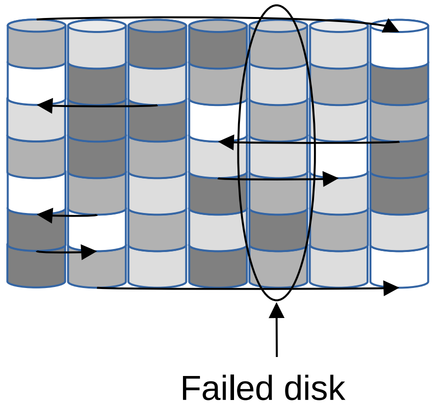
Belisama

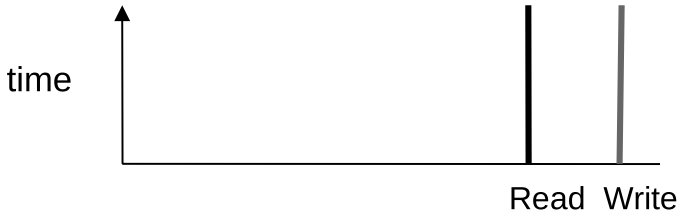


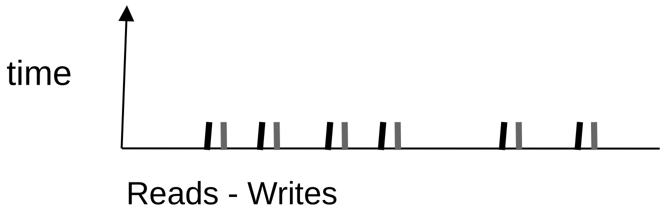


De-clustered RAID example









Rebuild activity confined to just a few disks – slow rebuild, disrupts user programmes

Rebuild activity spread across many disks, less disruption to user programmes

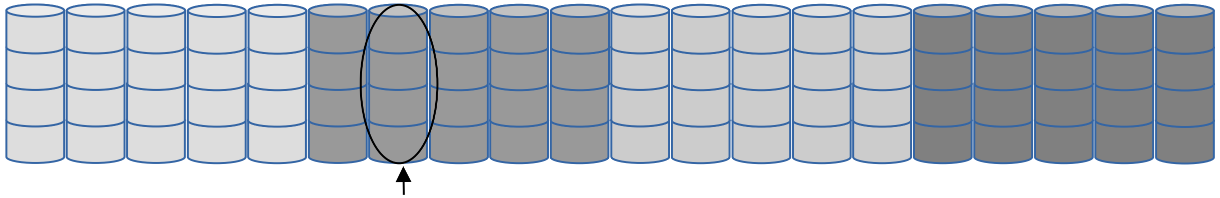
De-clustering – parallelism applied to spinning disks

Belisama

- Conventional RAID: Narrow data+parity arrays
  - Rebuild can only use the IO capacity of 4 (surviving) disks

20 disks (5 disks per 4 conventional RAID arrays)

4x4 RAID stripes
(data plus parity)

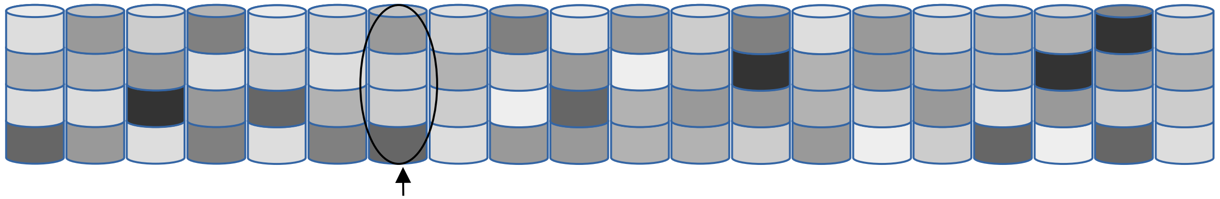

Failed disk

Striping across all arrays, all file accesses are throttled by array 2's rebuild overhead.

Declustered RAID: Data+parity distributed over all disks
  - Rebuild can use the IO capacity of all 19 (surviving) disks

20 disks in one de-clustered raid array

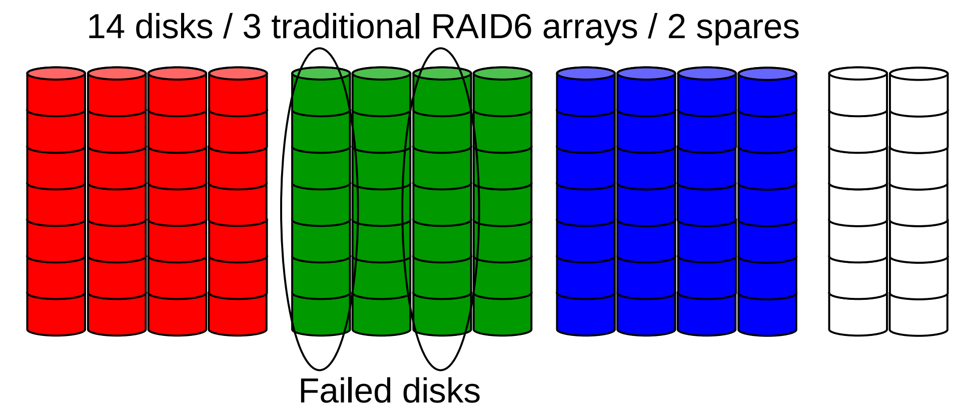
16 RAID stripes
(data plus parity)

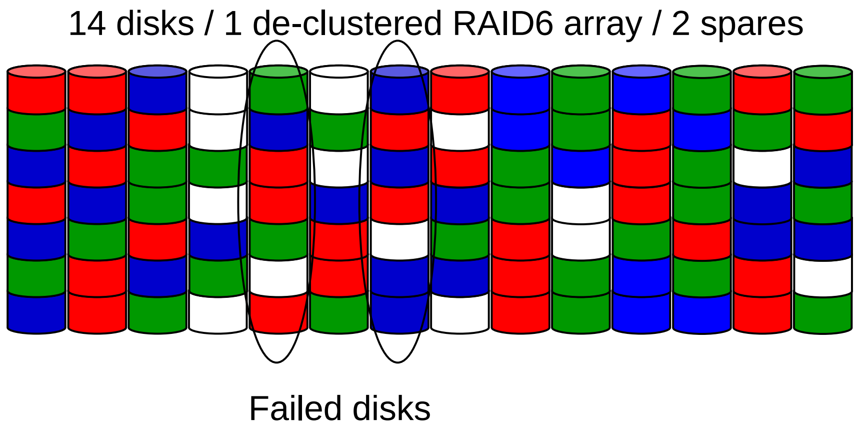

Failed disk

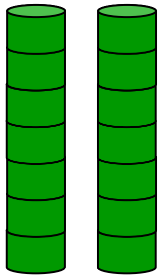
Load on files accesses are reduced by 4.8x(=19/4) during array rebuild.

De-clustered RAID 6 example

Belisama

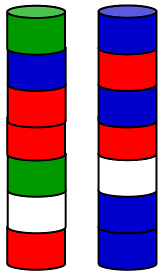






Number of failures per stripe		
Red	Green	Blue
	2	
	2	
	2	
	2	
	2	
	2	
	2	

7 stripes with 2 faults



Number of failures per stripe		
Red	Green	Blue
	1	1
1		1
1		1
2		
	1	
		1
1		1

1 stripe with 2 faults

Reference

- Internal Links
  - GPFS FAQ
    - http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.gpfs.doc/gpfs_faqs/gpfs_faqs.html
  - Todd's site
    - http://pokgsa.ibm.com/home/t/o/toddnev/web/
  - Developer wiki
    - http://gpfs.almaden.ibm.com/
- External web site
  - http://www-03.ibm.com/systems/clusters/software/gpfs/index.html
- IBM AIX sites, Firmware
  - supportsite.wss/brandmain?brandind=5000025
- Tools
  - Fix Level Recommendation Tool
    - http://www14.software.ibm.com/webapp/set2/flrt/home
  - IBM Pre-req tool
    - compare_report and subscription services

Reference

Belisama

- Redbooks
  - Implementing the IBM General Parallel File System (GPFS) in a Cross Platform Environment, SG24-7844-00
  - GPFS in the Cloud: Storage Virtualization with NPIV on IBM System p and IBM System Storage DS5300, REDP-4682-00
  - Deploying Oracle 10g RAC on AIX V5 with GPFS, SG24-7541-00
  - A Deployment Guide for Elastic Storage Object, REDP-5113-01
  - A Guide to the IBM Clustered Network File System, REDP-4400-01
  - IBM Spectrum Scale Best Practices for Genomics Medicine Workloads, April 2018, REDP-5479-01
- Useful youtube links:
  - GPFS GNR: https://www.youtube.com/watch?v=VvIgjVYPc_U

Session: s111035
Introduction to GPFS

¿ Questions ?

Thanks!

Your feedback about this session is very important to us.

Please remember to submit a survey

For further information….
Contact:

Antony (Red) Steel antony.steel@belisama.com.sg
+65 9789 6663

Power Systems