IBM TechU

Belisama

Introduction to Spectrum Scale (s203870)

Antony (Red) Steel
Belisama Pte. Ltd.

#include <std_disclaimer.h>
These notes have been prepared by an Australian, so beware of unusual spelling and pronunciation.
All comments regarding futures are probably nothing more than the imagination of the speaker and are IBM Confidential till after GA.

Thanks to:
Lin Feng Shen
Tom E O'Brien
Christopher Maestas

IBM Spectrum Scale

IBM

Spectrum Scale (GPFS)

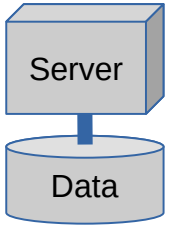
- Introduction
  - File System Types
  - History and Milestones
  - Usage scenarios
  - Key strengths and competition
- Understanding Spectrum Scale
  - Base Concepts
  - Network Shared Disks (NSD)
  - Blueprints & Daemons
- Get Started / How To Guide for AIX/Linux
  - Creating a Spectrum Scale Cluster
  - Network Shared Disk infrastructure
- Spectrum Scale features and functions
  - Availability, DR, AFM
  - New features, ECE, ESS, …..

Spectrum Scale / GPFS, what is it?

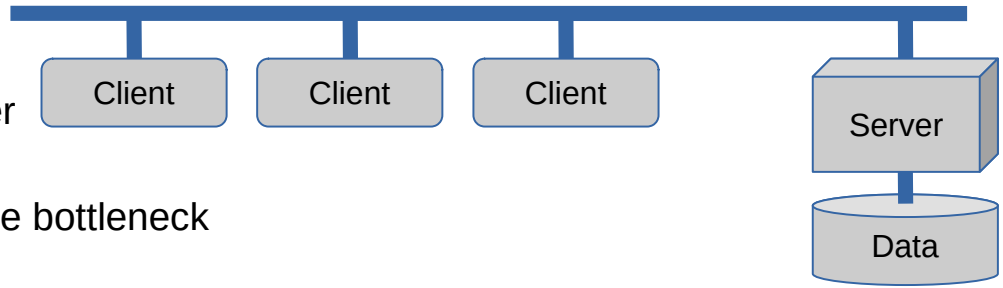
- What is it
  - Software from IBM that runs on AIX, Linux (p/x/z), Windows (Client / Server)
  - Serves data (file, object..) via GPFS “protocol” (Client Software), NFS, SMB, Swift
  - IBM also sells as
    - Appliance: ESS (older versions SoFS and SoNAS)
    - Solutions: DB2 PureScale, HPC, AI, SAP, 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

- Local file systems
  - File system 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, XFS, EXT3, EXT4, ..

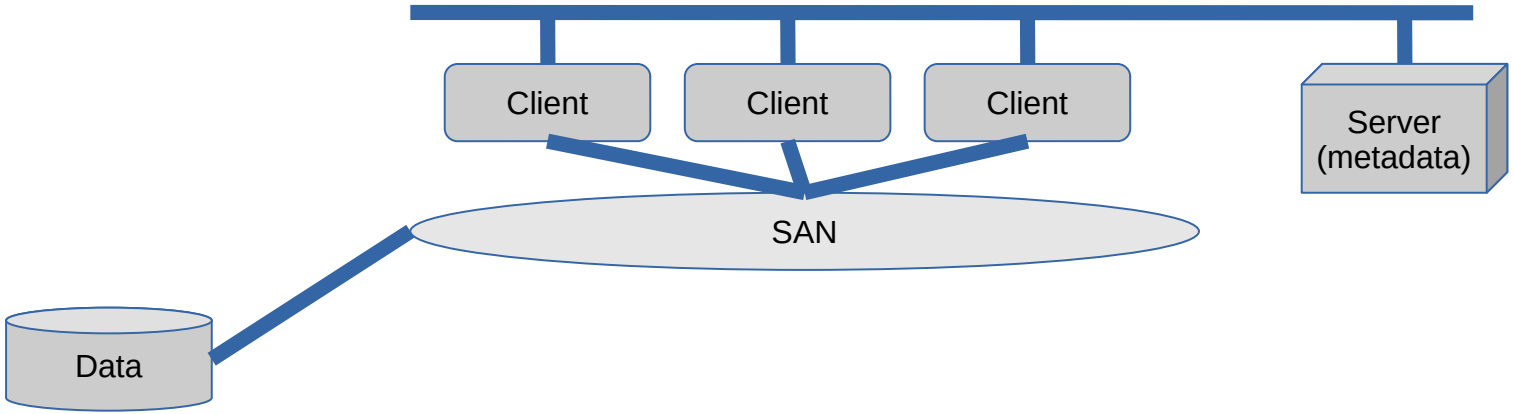


- Remote file systems
  - Data is placed on remote server
  - Metadata is maintained by remote server
  - File locking is done by remote server
  - Single server might become performance bottleneck
  - Examples: NFS v3/v4 (single server)



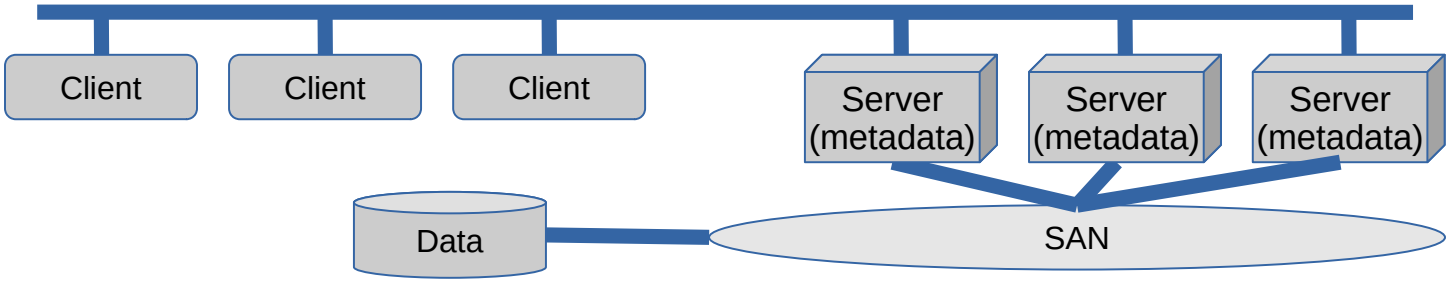
File System Types (cont)

- Shared file systems
  - 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; file system extensions (e.g. Veritas)

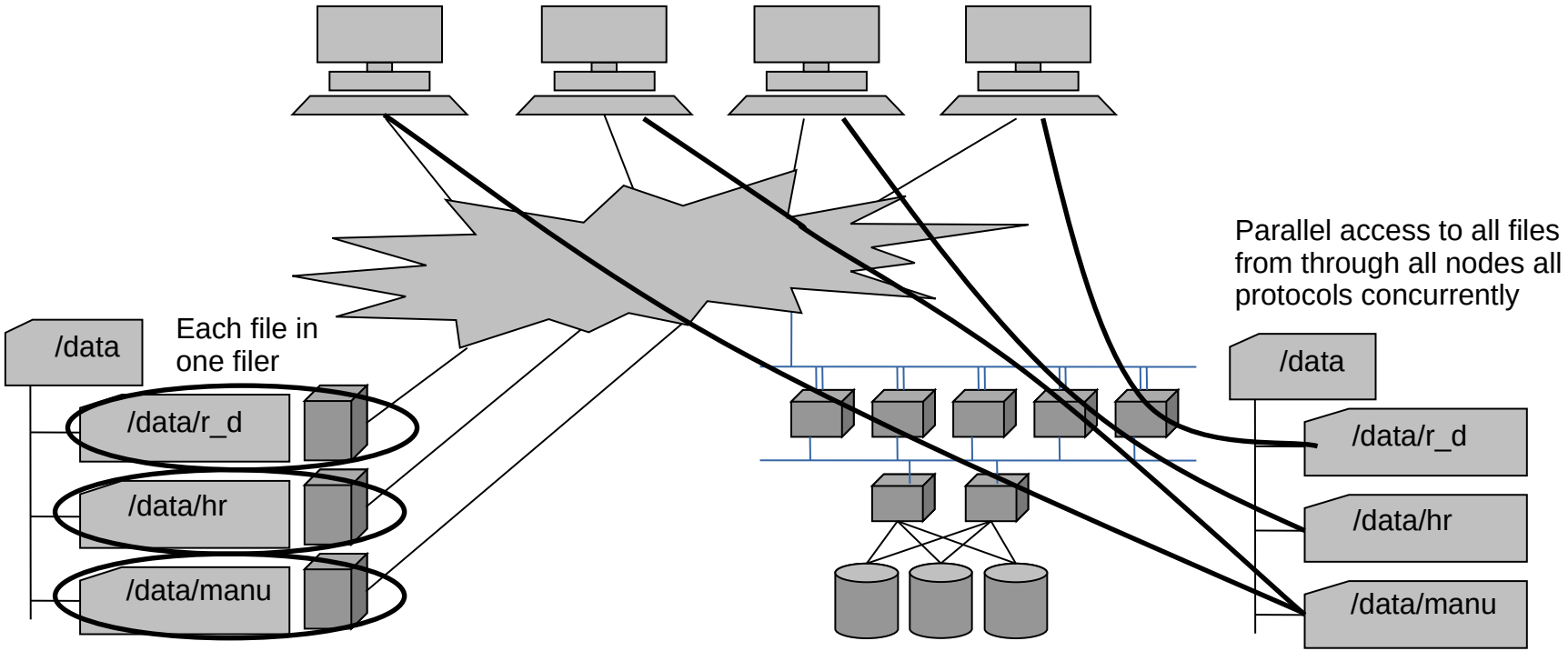


File Systems Types (cont)

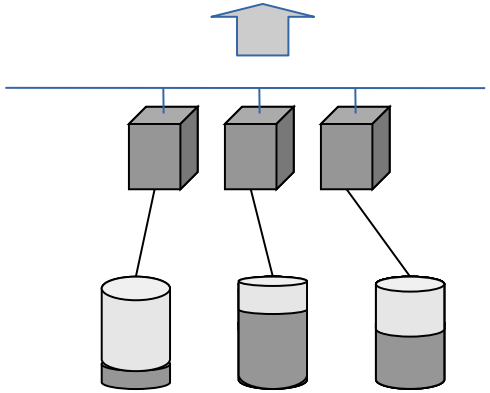
- Spectrum Scale / General Parallel File System
  - Data is striped across shared local disks (e.g. SAN) or NSD servers
  - Metadata is maintained by all servers in the cluster
  - File locking is distributed across the servers in the cluster
  - Excellent performance and scalability for large amounts of data
  - Very flexible configuration
  - Proven and mature high availability concepts, even for site disaster
  - Spectrum Scale 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)

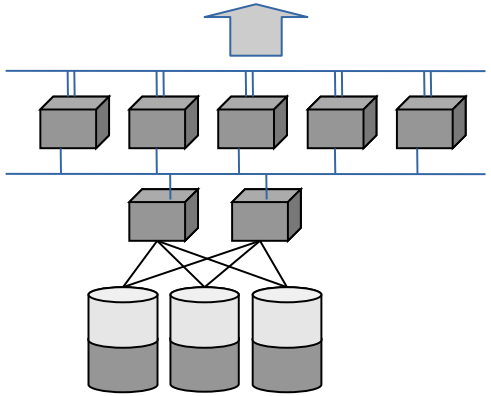


Spectrum Scale – Global Name Space



Spectrum Scale compared with Filers

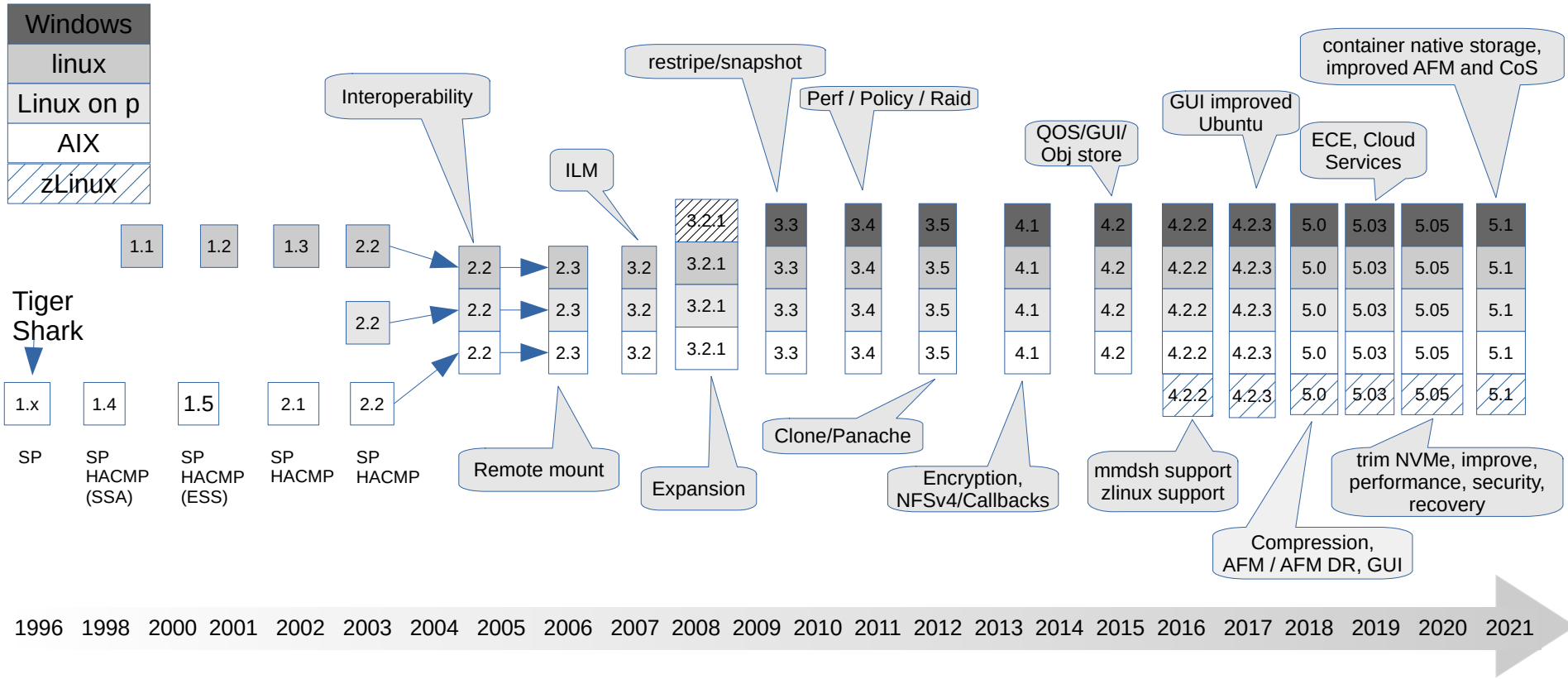




- Classic Fillers
  - “loved my first filler, so easy to manage, but when we installed the 20th….”
  - Individual management
  - Linear cost growth

- Spectrum Scale
  - Centrally managed
  - ILM part of Spectrum Scale
  - Easy growth and data migration

History and Milestones



Usage scenarios

- HPC - Scientific and technical environments
  - Research & HPC
  - Crash & NVH testing, CAE (Automotive and Aerospace)
  - Large Cluster (AIX, Linux, BlueGene/P)
  - WAN File system for Data Grids
- Commercial environments
  - Fast, scalable access to large amounts of file data
  - High Availability clusters (HA)
  - Db2 purescale
  - 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
  - Summit and Sierra
- SAP and oracle certified...
- Spectrum Protect integration (see Advanced admin guide)

Spectrum Scale clusters

- 1 to 8,192 Nodes supported
  - Tested up to 5,000 Linux nodes and 2,000 AIX Nodes
  - There are many Spectrum Scale 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
  - ppc64, ppc64le, x_86, x86_64 Distros: RHEL 5, 6 and 7 and SLES 10, 11 and 12 .. Ubuntu
  - Blue Gene (BG/L,BG/P)
  - Windows Windows 10; Windows Server 2016; Windows Server 2019
- Can run a mix of OS levels and a mix of AIX, Linux and Windows.
- There was a Management GUI 3.2/3.3 – gone 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 Spectrum Scale 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)

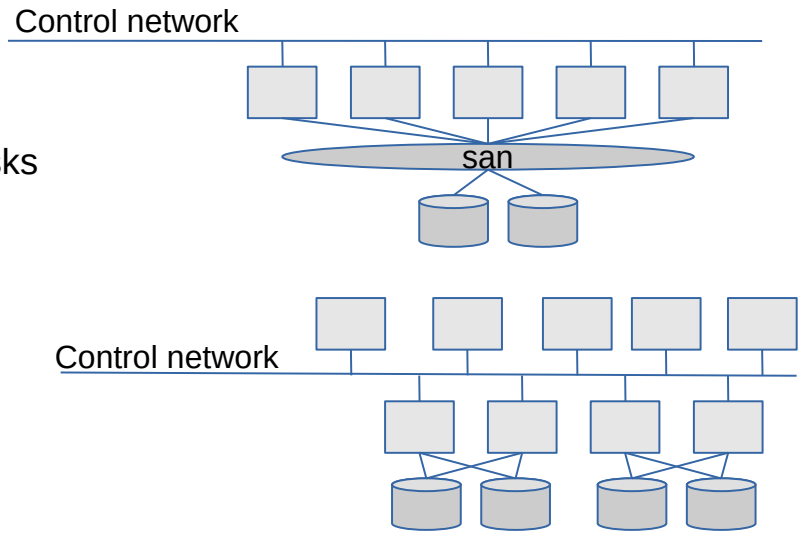
- Ease of use and robustness
  - Administration can be done from any node with simple commands
  - Online reconfiguration (adding and deleting disks and nodes)
  - High recover-ability and increased data availability
  - Information Life-cycle Management (ILM)
- Scalability and performance
  - Scalability 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 spectrum scale clusters
  - Improved file serving for Network File system (NFS) v4 functions and performance

Limitations

- GPFS 2.3, or later, architectural file system size limit
  - 2*99 bytes
  - Current tested limit over 900PB
- Total number of files per file system
  - 2*63 (over 40 billion tested) Note: GPFS 3.3 and earlier was 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 file system first created)
- Maximum number of snapshots
  - 256

Base Concepts

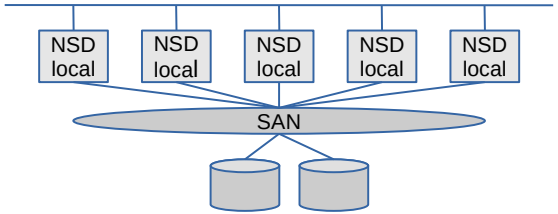
- Technical concepts
  - Shared Disks
    - All data and metadata on globally accessible block storage
  - Wide Striping
    - All data and metadata can be striped across all disks
    - Files striped block by block across all disks
    - … for throughput and load balancing
  - Distributed Metadata
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

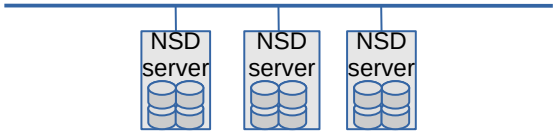


Principle: scalability through parallelism and autonomy

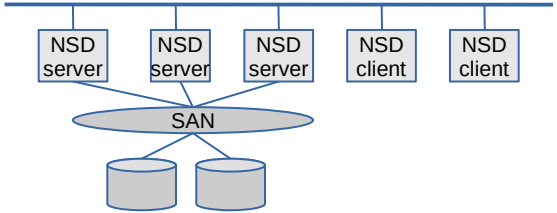
Network shared disks (NSD) architecture

- 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

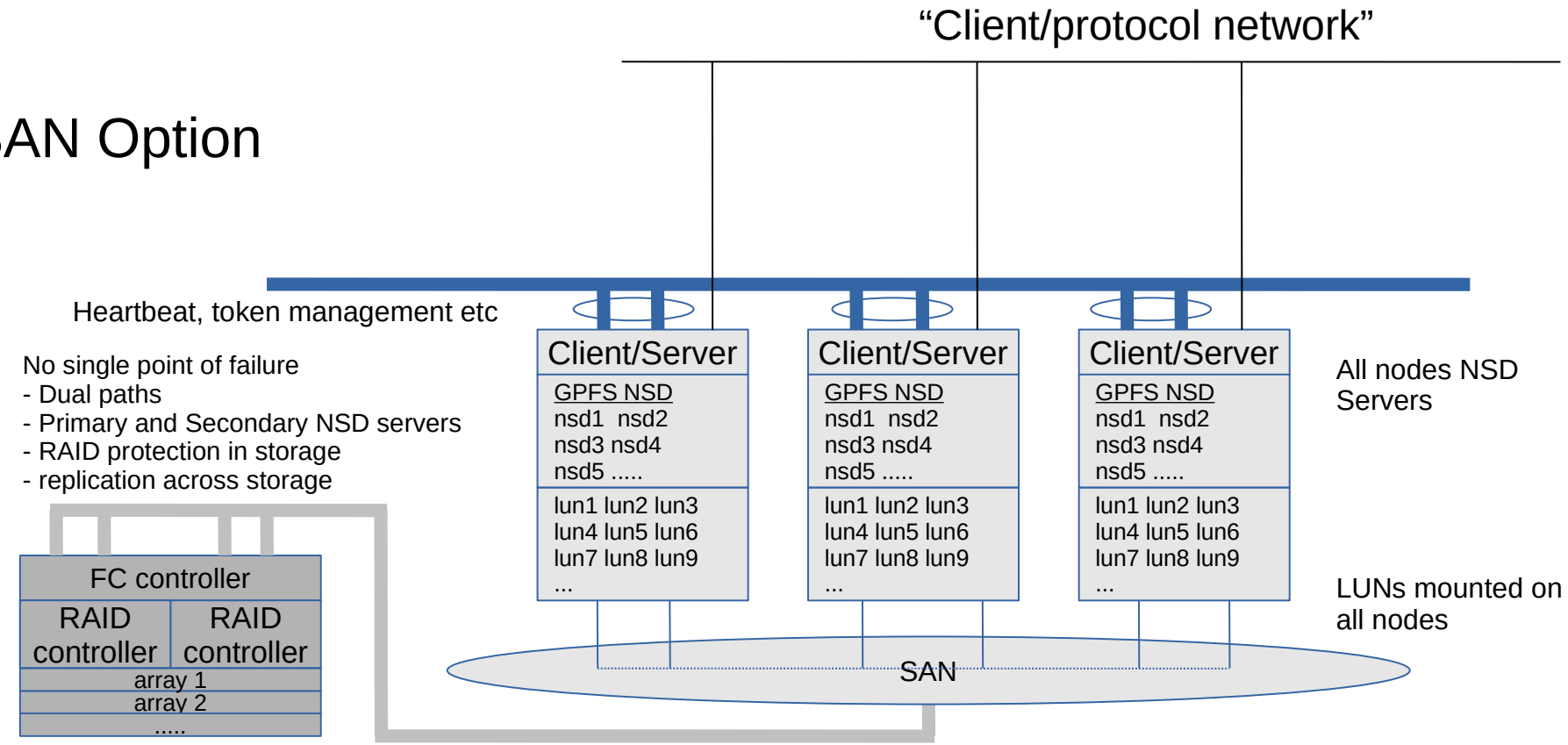


- FPO / Shared nothing
  - copies of data and metadata spread across the servers/storage for availability and performance

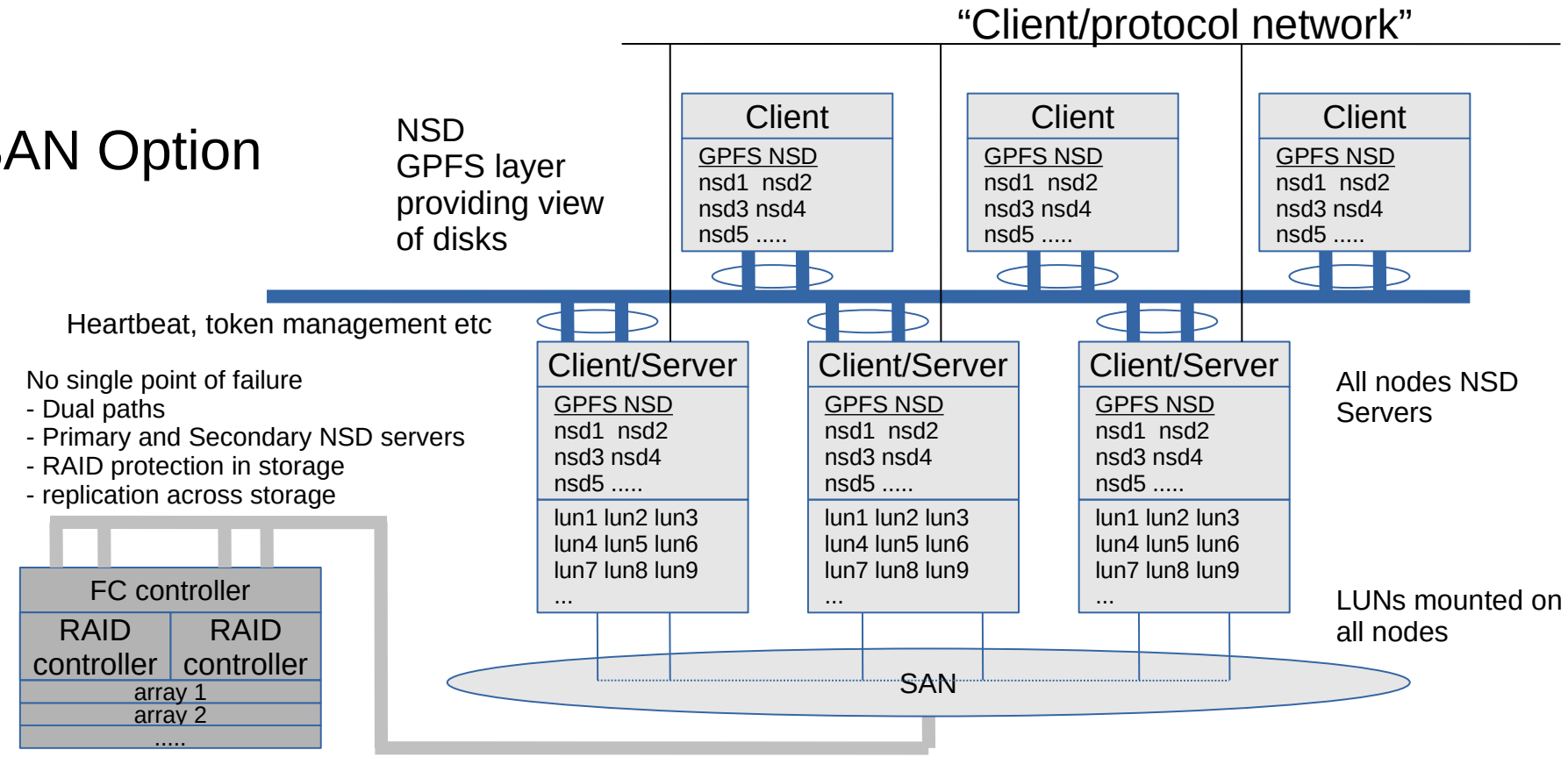
Base Concepts (cont)

- Spectrum Scale Components
  - Nodes
    - Spectrum Scale 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 or Infiniband 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 Spectrum Scale 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 Spectrum Scale, 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 Spectrum Scale, you may give an NSD name to:
      - Physical disks
      - Representations of physical disks (such as LUNs)

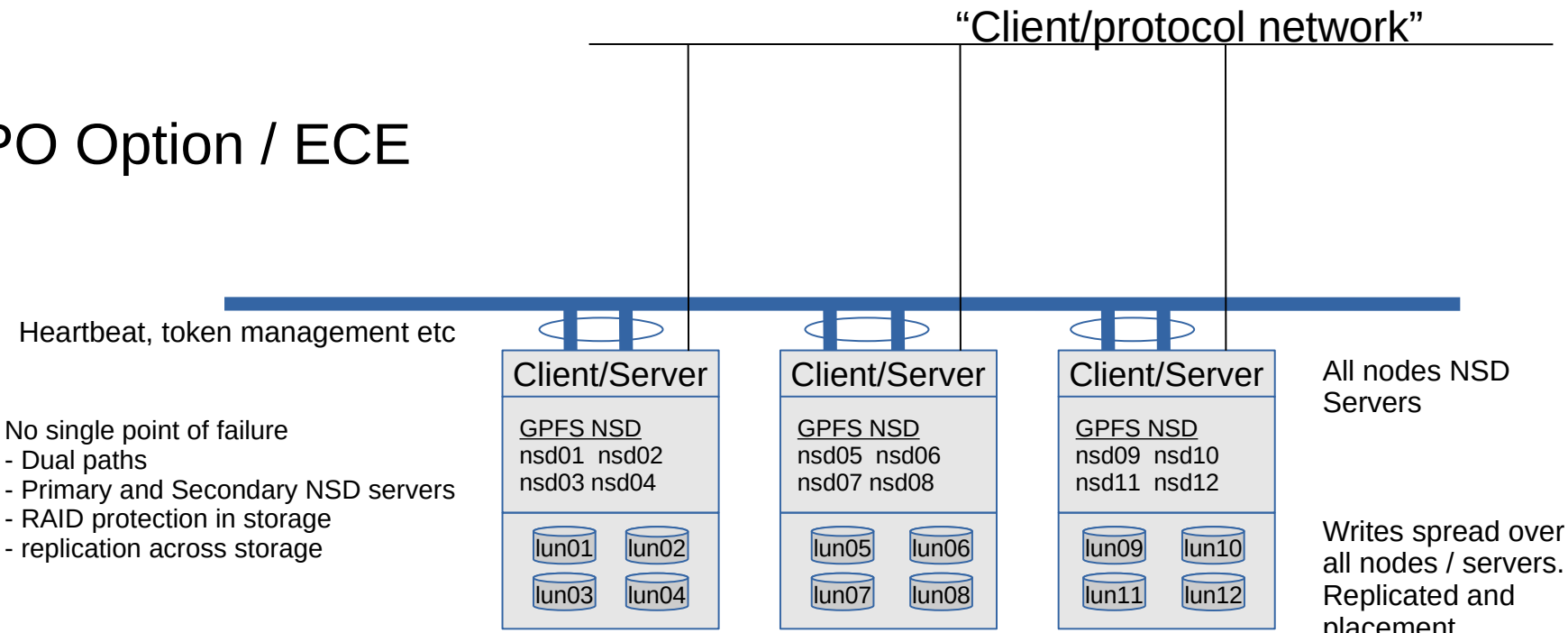
SAN Option



SAN Option



FPO Option / ECE



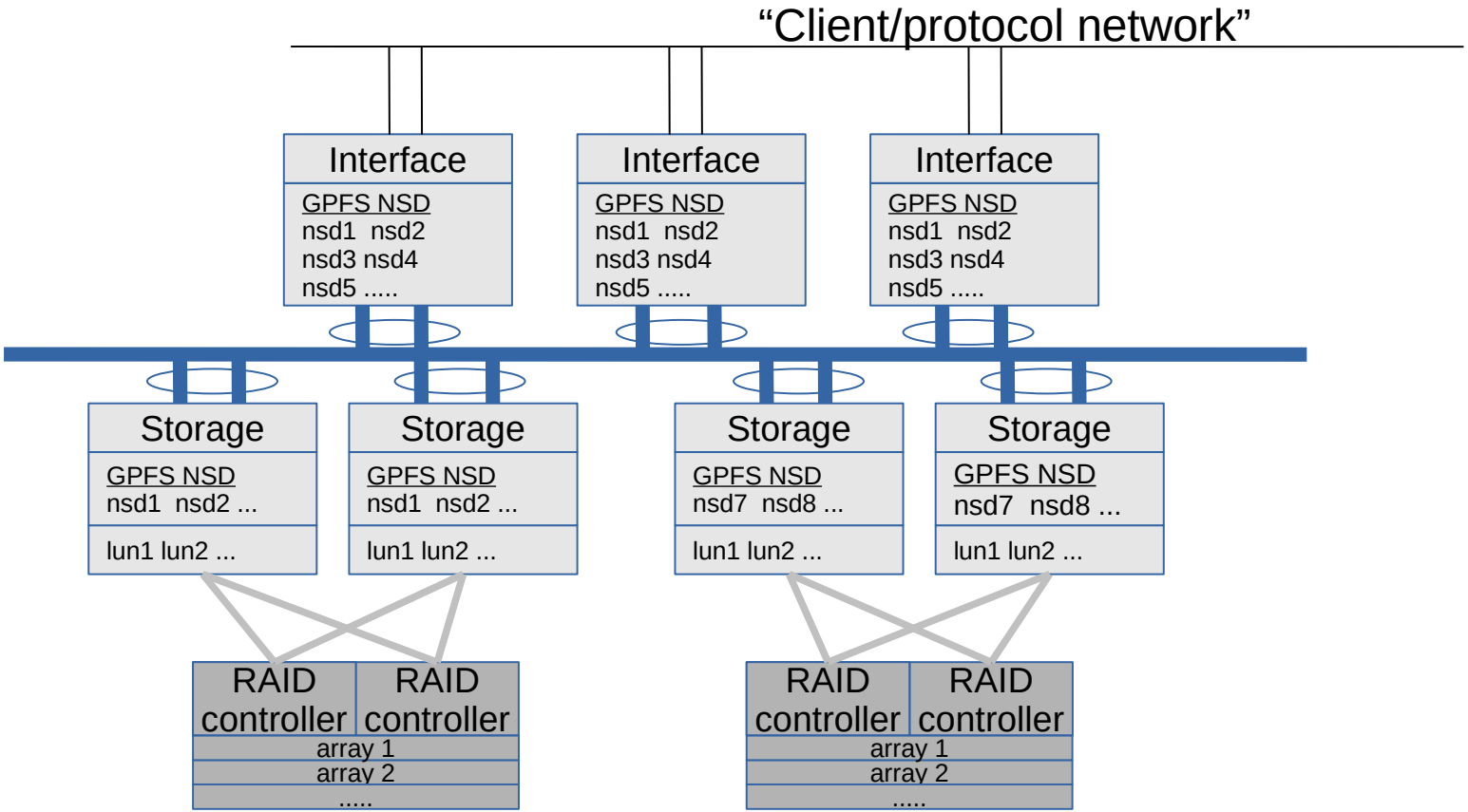
No single point of failure
- Dual paths
- Primary and Secondary NSD servers
- RAID protection in storage
- replication across storage

All nodes NSD Servers

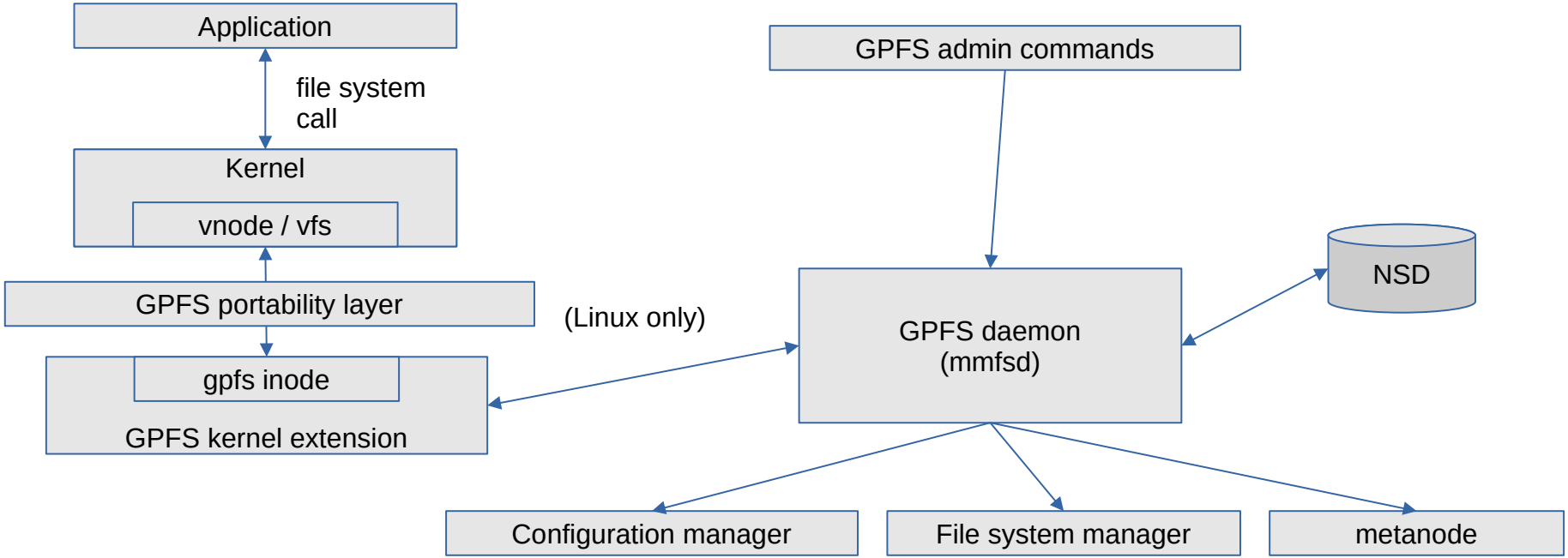
Writes spread over all nodes / servers. Replicated and placement optimised by proximity to the process

Base Concepts (cont) - the ESS view of Spectrum Scale

ESS

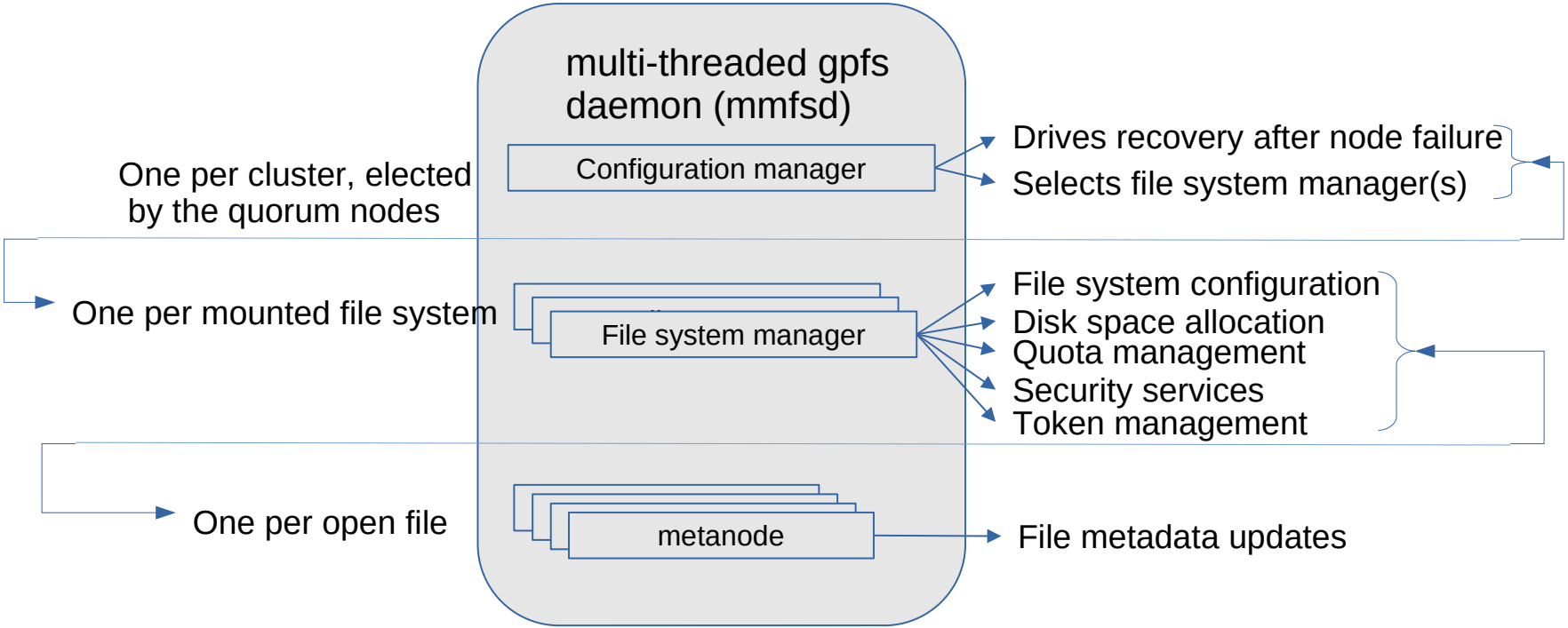


Base Concepts (cont) – the GPFS structure



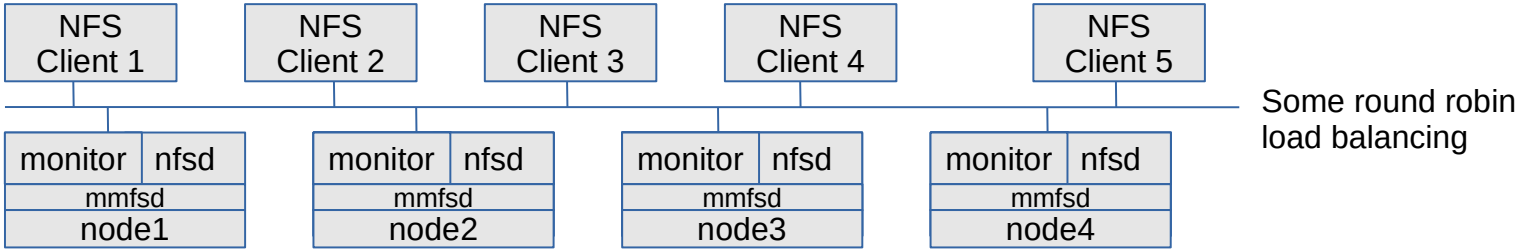
Base Concepts (cont)

- GPFS daemon (mmfsd) roles



Base Concepts (cont)

- 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 Spectrum Scale cluster

- Plan a Spectrum Scale 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 Spectrum Scale cluster

Planning considerations

- Nodes
  - Sizing
  - Number of nodes to provide throughput
  - Node quorum considerations (or tiebreaker disks for very small clusters)
  - Small odd number of most reliable nodes spread across the infrastructure
- File systems
  - Number of file systems
    - 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?
      - < 263 files, with Size limit 299 Bytes -> Current tested size order for 100s PB
    - Create file systems to support performance requirements
      - Disk type differences can be addressed using Storage Pools

Planning considerations (cont)

- Block size
  - 64KB block size 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 file systems to application block size:
    - /gpfs1 – 64K block size
    - /gpfs2 – 256K block size
  - Numbers of file system replicas
- Segments = 1/32 of a block
  - The smallest amount of disk allocated by Spectrum Scale
  - variable block size introduced in version 5
    - by default file system block size 4M with a sub-block size of 8K
    - select from 64 to 2048 sub-blocks of 8K or 16K

Planning considerations (cont)

- 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.
  - For efficiency generally use a smaller number of file systems
  - 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 or portion of the 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

Still further planning considerations

- Application - number of file systems
  - Different than planning local file systems
    - Fewer file systems
    - Multiple applications can often share a file system
  - Separate clusters – separate security contexts and 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

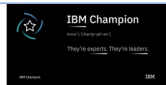
Still further planning considerations

- 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

Operating system considerations

- Supported on Power, x86 and Z
- Supported on AIX, Linux (ppc64le, x86, zLinux) and Windows
- For Linux
  - Compile the General Portability Layer
  - From 5.0.5 supports SELinux modes enforcing or permissive with SELinux policy set to a targeted policy.
- For Windows
  - Some limitations as NSD Server
  - Prerequisites such as Cygwin

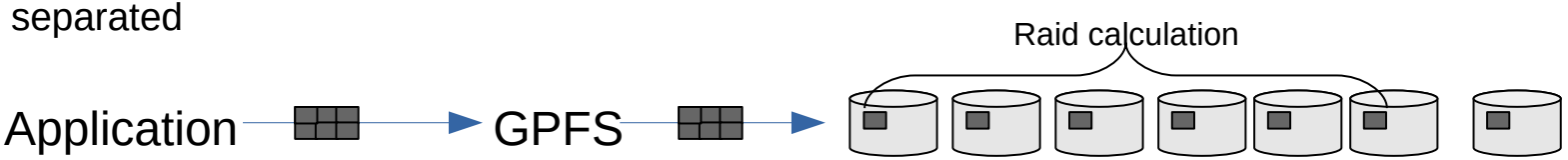
See Spectrum Scale FAQs for latest requirements



Storage considerations

- 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)
    - Plan for number of replicas
  - Metadata and data can be shared or separated

- 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 – separate by cost \ perfomance of underlying storage
    - tune LUN config for access type / access size

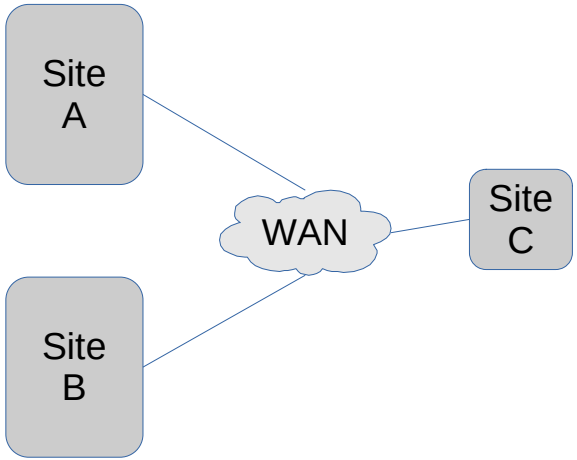


Network considerations

- 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

Spectrum Scale 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, file system descriptor
    - Serves as tie breaker if one of the other sites becomes inaccessible



Inside GPFS

- A GPFS cluster
- Nodes, disks and file systems
- Using the file system
- Monitoring the cluster

- 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

- 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

- 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 old configuration method with a primary and secondary configuration server
    - -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

- 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

```
GPFS cluster information
========================
  GPFS cluster name:         PVS.gpfs1_priv
  GPFS cluster id:           5936389947168084999
  GPFS UID domain:           PVS.gpfs1_priv
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp
  Repository type:           CCR

 Node  Daemon node name  IP address      Admin node name  Designation
-----------------------------------------------------------------------
   1   gpfs1_priv        192.168.50.156  gpfs1_priv       quorum-manager
   2   gpfs2_priv        192.168.50.193  gpfs2_priv       quorum-manager
   3   gpfs3_priv        192.168.50.137  gpfs3_priv       quorum-manager
```

View information on the GPFS configuration

- 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
Configuration data for cluster PVS.gpfs1_priv:
----------------------------------------------
clusterName PVS.gpfs1_priv
clusterId 5936389947168084999
autoload no
dmapiFileHandleSize 32
minReleaseLevel 5.1.1.0
ccrEnabled yes
cipherList AUTHONLY
sdrNotifyAuthEnabled yes
adminMode central
```

```
maxFilesToCache 4000
maxStatCache 1000
failureDetectionTime 35
maxMBpS 2048
unmountOnDiskFail no
allowSambaCaseInsensitiveLookup no
enableLowspaceEvents no
cipherList AUTHONLY
pagepool 1G
dmapiDataEventRetry 2
verifyGpfsReady no
```

Important tuning parameters

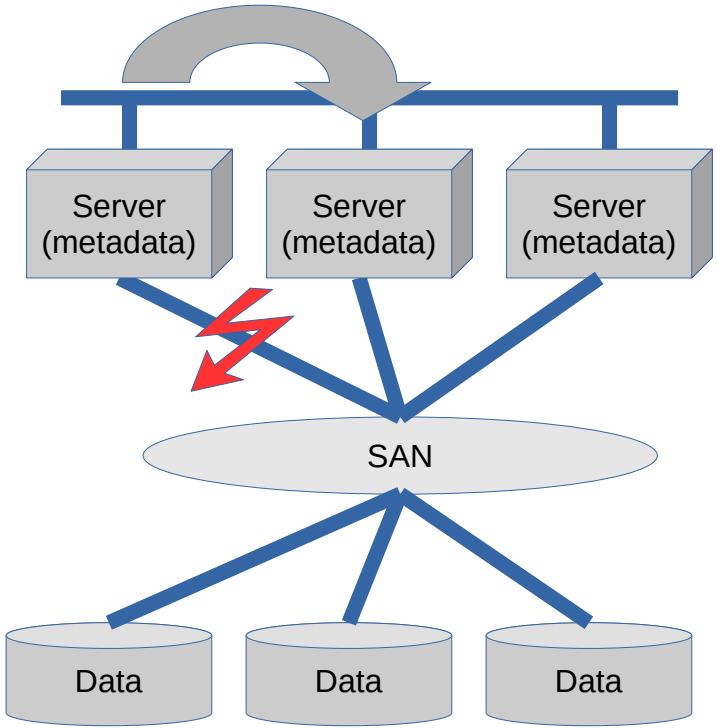
- GPFS use of memory
  - Two areas of memory
  - Pinned (pagepool) – used to store user data and file system metadata to support I/O operations
  - Not Pinned – two levels of cache for storing file metadata
  - Pagepool
    - The pagepool mechanism allows GPFS to implement read as well as write requests 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

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
  - 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
  - Size of the shared memory segment (kernel and mmfs daemon) used by GPFS
- 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)

Important tuning parameters (cont)

- 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 file system (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.
- For the following two file system settings, remember that stat() calls are expensive, so if your application vendor has concerns, leave at default
  - 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.
  - 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.

Important tuning parameters (cont)

- 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

- 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
Thu Aug 26 05:13:42 EDT 2021: 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 -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state   Remarks
------------------------------------------------------------------------------------
       1      gpfs1_priv         2        3          3       active       quorum node
       2      gpfs2_priv         2        3          3       active       quorum node
       3      gpfs3_priv         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
```

Stopping the cluster

- 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
Thu Aug 26 06:21:22 EDT 2021: mmshutdown: Starting force unmount of GPFS file systems
Thu Aug 26 06:21:27 EDT 2021: mmshutdown: Shutting down GPFS daemons
Thu Aug 26 06:21:35 EDT 2021: 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 file system

Create a NSD

- 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: device=DiskName
   nsd=NsdName
   servers=ServerList
   usage={dataOnly | metadataOnly | dataAndMetadata | descOnly | localCache}
   failureGroup=FailureGroup
   pool=StoragePool
   thinDiskType={no | nvme | scsi | auto}
```

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.

- Once mmcrnsd has completed
  - the NSDs are usable in GPFS
  - The NSD descriptor file can also be used as an input file for other commands

NSD descriptor file

```
%nsd: device=DiskName
  nsd=NsdName
  servers=ServerList
  usage={dataOnly | metadataOnly | dataAndMetadata |
         descOnly | localCache}
  failureGroup=FailureGroup
  pool=StoragePool
  thinDiskType={no | nvme | scsi | auto}
```

- Records in the NDS descriptor file are:
  - device
    - the block device name appearing in /dev for the disk
  - nsd
    - name for the NSD to be created (default gpfsNNnsd)
  - servers (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
  - usage
    - 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
      - localCache can be used for locally caching remote data
  - failureGroup
    - A number identifying the failure group.
    - Used for replication, each replica will be stored in a different failure group – discussed later under HA

NSD descriptor file (cont)

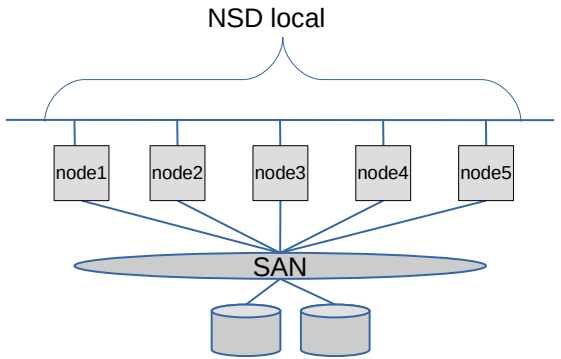
- Records in the NDS descriptor file are:
  - pool
    - Specifies the name of the storage pool that the NSD is assigned to and is used to group like disks for ILM.
  - thinDiskType
    - Specifies the space reclaim disk type
      - no - the disk device supports space reclaim
        - This is the default
      - nvme – the disk device is a TRIM capable NVMe device that supports the mmreclaimspace command
      - scsi - the disk device is a thin provisioned SCSI disk that supports the mmreclaimspace command
      - auto – for nvme and scsi devices to let Spectrum Scale detect actual type
        - Recommended use the actual type

```
%nsd: device=DiskName
  nsd=NsdName
  servers=ServerList
  usage={dataOnly | metadataOnly | dataAndMetadata |
         descOnly | localCache}
  failureGroup=FailureGroup
  pool=StoragePool
  thinDiskType={no | nvme | scsi | auto}
```

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.

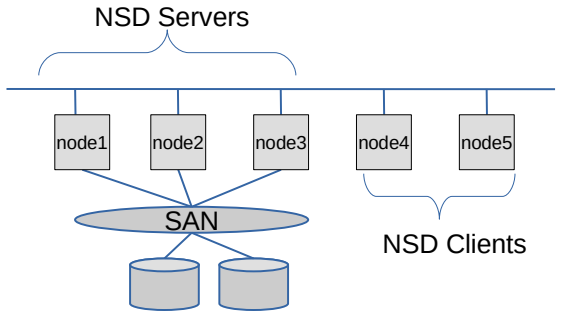
```
%nsd:
   nsd=nsd01
   usage=dataAndMetadata
   failureGroup=-1
   pool=System
   device=/dev/dm-0
%nsd:
   nsd=nsd02
   usage=dataAndMetadata
   failureGroup=-1
   pool=System
   device=/dev/dm-1
```



Create a NSD descriptor file for NSD over LAN

- 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 ./nsd.txt
mmcrnsd: Processing disk dm-0
mmcrnsd: Processing disk dm-1
mmcrnsd: Processing disk dm-2
mmcrnsd: Processing disk dm-3
mmcrnsd: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.

# mmlsnsd

 File system   Disk name       NSD servers
---------------------------------------------------------------
 (free disk)   nsd1            gpfs1_priv,gpfs2_priv,gpfs3_priv
 (free disk)   nsd2            gpfs2_priv,gpfs3_priv,gpfs1_priv
 (free disk)   nsd3            gpfs3_priv,gpfs1_priv,gpfs2_priv
 (free disk)   nsd4            gpfs1_priv,gpfs2_priv,gpfs3_priv
```

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

mmlsd information

```
# mmlsnsd -X

 Disk name       NSD volume ID      Device          Devtype  Node name or Class       Remarks
---------------------------------------------------------------------------------------------------
 nsd1            C0A88AFE612774D6   /dev/dm-0       dmm      gpfs1_priv               server node
 nsd1            C0A88AFE612774D6   /dev/dm-2       dmm      gpfs2_priv               server node
 nsd1            C0A88AFE612774D6   /dev/dm-3       dmm      gpfs3_priv               server node
 nsd2            C0A88AFA612774D8   /dev/dm-2       dmm      gpfs1_priv               server node
 nsd2            C0A88AFA612774D8   /dev/dm-1       dmm      gpfs2_priv               server node
 nsd2            C0A88AFA612774D8   /dev/dm-1       dmm      gpfs3_priv               server node
 nsd3            C0A88AFB612774DA   /dev/dm-1       dmm      gpfs1_priv               server node
 nsd3            C0A88AFB612774DA   /dev/dm-0       dmm      gpfs2_priv               server node
 nsd3            C0A88AFB612774DA   /dev/dm-2       dmm      gpfs3_priv               server node
 nsd4            C0A88AFE612774DC   /dev/dm-3       dmm      gpfs1_priv               server node
 nsd4            C0A88AFE612774DC   /dev/dm-3       dmm      gpfs2_priv               server node
 nsd4            C0A88AFE612774DC   /dev/dm-0       dmm      gpfs3_priv               server node
```

device mapper multipath

Create a GPFS file system

- 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 file system. However with the new stanza format, the same file can be used to create NSDs and the file system. 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 file system (cont)

- mmcrfs creates a file system
  - Need to specify mountpoint, devicename and disks
  - Many other options!

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

The following disks of gpfs0 will be formatted on node gpfs-store-3:
    nsd1: size 51200 MB
    nsd2: size 51200 MB
    nsd3: size 51200 MB
    nsd4: size 51200 MB
Formatting file system ...
Disks up to size 441.49 GB can be added to storage pool system.
Disks up to size 441.49 GB can be added to storage pool platinum.
Creating Inode File
Creating Allocation Maps
Creating Log Files
Clearing Inode Allocation Map
Clearing Block Allocation Map
Formatting Allocation Map for storage pool system
Formatting Allocation Map for storage pool platinum
Completed creation of file system /dev/gpfs0.
mmcrfs: Propagating the cluster configuration data to all
  affected nodes.  This is an asynchronous process.
```

Create a GPFS file system (cont)


```
flag                value                    description
------------------- ------------------------ -----------------------------------
 -f                 8192                     Minimum fragment (subblock) size in bytes
 -i                 4096                     Inode size in bytes
 -I                 32768                    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                 all                      ACL semantics in effect
 -n                 32                       Estimated number of nodes that will mount file system
 -B                 4194304                  Block size
 -Q                 none                     Quotas accounting enabled
                    none                     Quotas enforced
                    none                     Default quotas enabled
 --perfileset-quota no                       Per-fileset quota enforcement
 --filesetdf        no                       Fileset df enabled?
 -V                 25.00 (5.1.1.0)          File system version
--create-time       Thu Aug 26 07:14:03 2021 File system creation time
 -z                 no                       Is DMAPI enabled?
 -L                 33554432                 Logfile size
 -E                 yes                      Exact mtime mount option
 -S                 relatime                 Suppress atime mount option
 -K                 whenpossible             Strict replica allocation option
 --fastea           yes                      Fast external attributes enabled?
 --encryption       no                       Encryption enabled?
 --inode-limit      206848                   Maximum number of inodes
 --log-replicas     0                        Number of log replicas
 --is4KAligned      yes                      is4KAligned?
 --rapid-repair     yes                      rapidRepair enabled?
 --write-cache-threshold 0                   HAWC Threshold (max 65536)
 --subblocks-per-full-block 512              Number of subblocks per full block
 -P                 system;platinum          Disk storage pools in file system
 --file-audit-log   no                       File Audit Logging enabled?
 --maintenance-mode no                       Maintenance Mode enabled?
 -d                 nsd1;nsd2;nsd3;nsd4      Disks in file system
 -A                 yes                      Automatic mount option
 -o                 none                     Additional mount options
 -T                 /gpfs                    Default mount point
 --mount-priority   0                        Mount priority
```

Create a GPFS file system (cont)

- Now we just need to mount the file system
  - mmmount [file system] -a
  - option to mount on some nodes, use -N nodelist

```
# mmmount gpfs0 -a
Thu Aug 26 07:25:04 EDT 2021: mmmount: Mounting file systems ...

# mmdsh df /gpfs
gpfs1_priv:  Filesystem     1K-blocks      Used Available Use% Mounted on
gpfs1_priv:  gpfs0          104857600 104857600         0 100% /gpfs
gpfs2_priv:  Filesystem     1K-blocks      Used Available Use% Mounted on
gpfs2_priv:  gpfs0          104857600 104857600         0 100% /gpfs
gpfs3_priv:  Filesystem     1K-blocks   Used Available Use% Mounted on
gpfs3_priv:  gpfs0          104857600 147456 104710144   1% /gpfs
```

Disk attributes

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

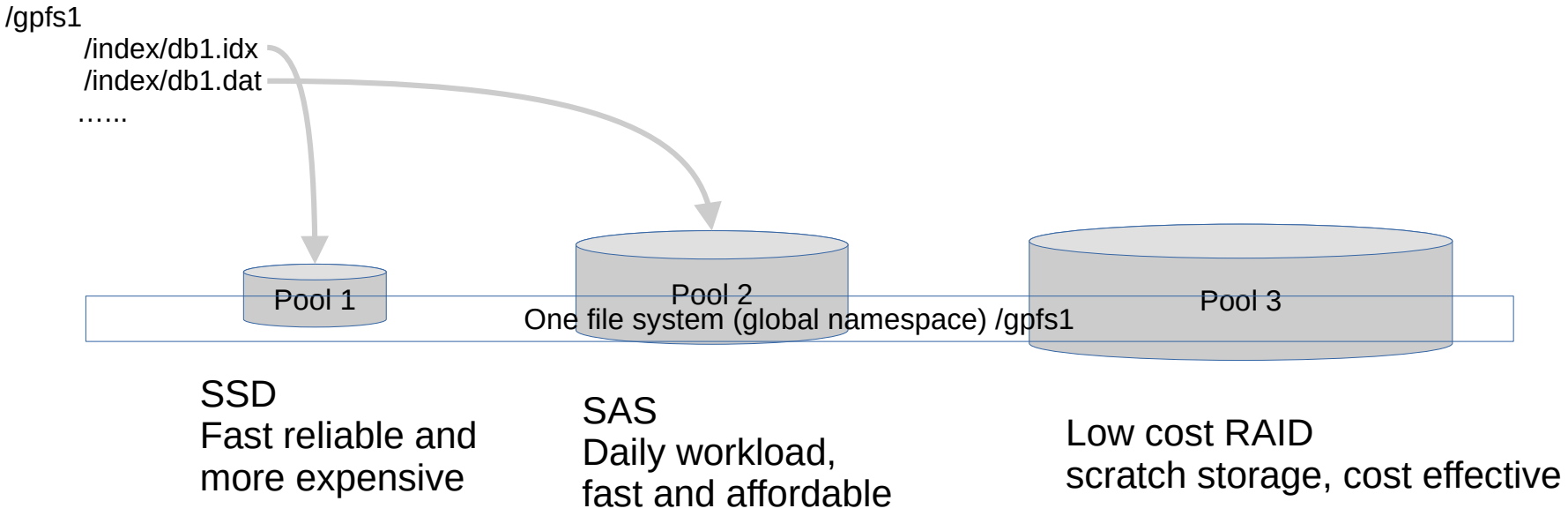
```
# mmlsdisk gpfs0
disk         driver   sector     failure holds    holds                            storage
name         type       size       group metadata data  status        availability pool
------------ -------- ------ ----------- -------- ----- ------------- ------------ ------------
nsd1         nsd         512           1 yes      no    ready         up           system
nsd2         nsd         512          10 yes      no    ready         up           system
nsd3         nsd         512           1 no       yes   ready         up           platinum
nsd4         nsd         512          10 no       yes   ready         up           platinum
```

GPFS features and functions - policies

- How GPFS handles storage
  - Storage Pools
  - Filesets
  - Policies and placement
  - Quotas
  - Snapshots
  - Data Management API (DMAPI)
  - GPFS and hierarchical storage management (HSM)
  - Remote mount capabilities

Storage pools

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



Storage pools (cont)

- 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 File system
- Each disk has this attribute in its disk descriptor
  - At creation time
  - At the time the disk is added to the file system
- Files are assigned to storage
  - At creation time
  - Attributes of the file, match the rules of an active policy

Using storage pools

- 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;platinum          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
immutable:            no
appendOnly:           no
flags:
storage pool name:    platinum
fileset name:         root
snapshot name:
creation time:        Thu Aug 26 07:29:43 2021
Misc attributes:      ARCHIVE
Encrypted:            no
```

Using storage pools (cont)

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

```
# mmdf gpfs0 -P system
disk                disk size  failure holds    holds           free in KB          free in KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 441.49 GB)
nsd1                 52428800        1 yes      no         51392512 ( 98%)         10104 ( 0%)
nsd2                 52428800       10 yes      no         51384320 ( 98%)         10840 ( 0%)
                -------------                         -------------------- -------------------
(pool total)        104857600                             102776832 ( 98%)         20944 ( 0%)
```

Using storage pools (cont)


```
gpfs1:/gpfs# mmdf gpfs0
disk                disk size  failure holds    holds           free in KB          free in KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: system (Maximum disk size allowed is 441.49 GB)
nsd1                 52428800        1 yes      no          51392512 ( 98%)         10104 ( 0%)
nsd2                 52428800       10 yes      no          51384320 ( 98%)         10840 ( 0%)
             -------------                                -------------------- -------------------
(pool total)        104857600                                102776832 ( 98%)         20944 ( 0%)

Disks in storage pool: platinum (Maximum disk size allowed is 441.49 GB)
nsd3                 52428800        1 no       yes         52355072 (100%)          8056 ( 0%)
nsd4                 52428800       10 no       yes         52355072 (100%)          8056 ( 0%)
             -------------                                -------------------- -------------------
(pool total)        104857600                                104710144 (100%)         16112 ( 0%)

             =============                                ==================== ===================
(data)              104857600                                104710144 (100%)         16112 ( 0%)
(metadata)          104857600                                102776832 ( 98%)         20944 ( 0%)
             =============                                ==================== ===================
(total)             209715200                                207486976 ( 99%)         37056 ( 0%)

Inode Information
-----------------
Number of used inodes:           4039
Number of free inodes:         202809
Number of allocated inodes:    206848
Maximum number of inodes:      206848
```

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 file systems (i.e. failure groups)

Filesets

- A fileset is a subtree of a file system namespace that behaves very much like an independent file system.
  - Can be linked (mounted) on any point in the filesystem
- Filesets provide a means of partitioning a file system for administrative purposes, with finer granularity.
  - Can be used to define quotas (data blocks, inodes)
  - Define quotas (disk blocks and inode) at fileset level (`-j` flag in quota commands)
  - Apply policy rules to specific filesets
- 10 000 filesets per File system (3.4 and above)
- GPFS 3.5 introduced dependent and independent filesets
  - Dependent
    - Use inodes / space / snapshots from ‘global’ file system
  - Independent
    - Own inodes and snapshots, space from ‘global’ file system
- The root fileset always exists for each file system
- mmlsattr –L shows fileset membership of a file

Filesets (cont)

- filesets are attached or linked to a *junction* in the file system. A *junction* is a special entry that connects a source to the root directory of the target fileset. Only one fileset can be the target of a junction and it appears as a directory.
  - use mmlinkfileset to link a fileset
- Use mmunlinkfileset to unlink a fileset
  - Makes files inaccessible, but still exist
- Namespace in a fileset is a single connected tree, one root directory and no entry points such as hard links from other filesets.
  - Hard links cannot cross fileset boundaries, symbolic links can
- Symbolic links can be used
- Filesets and storage pools not specially related

Filesets (cont)

- Administration
  - Creating: mmcrfileset
    - Character string < 256
    - Unique within a file system
    - 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

- Policy: Set of rules defining the life cycle of user defined data (SQL like language)
- 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 File system: 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
    - SQL_exprn for example: 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 file system
  - mmlspolicy
    - List policies and information
  - mmrestripefile
    - Re-balance files within storage pools or for storage pools

Quotas

- The GPFS quota system helps you to control the allocation of files and data blocks in a file system. Quotas can be defined for:
  - Individual users
  - Groups of users
  - Individual filesets
- By default apply across the whole file system, but can be limited by fileset.

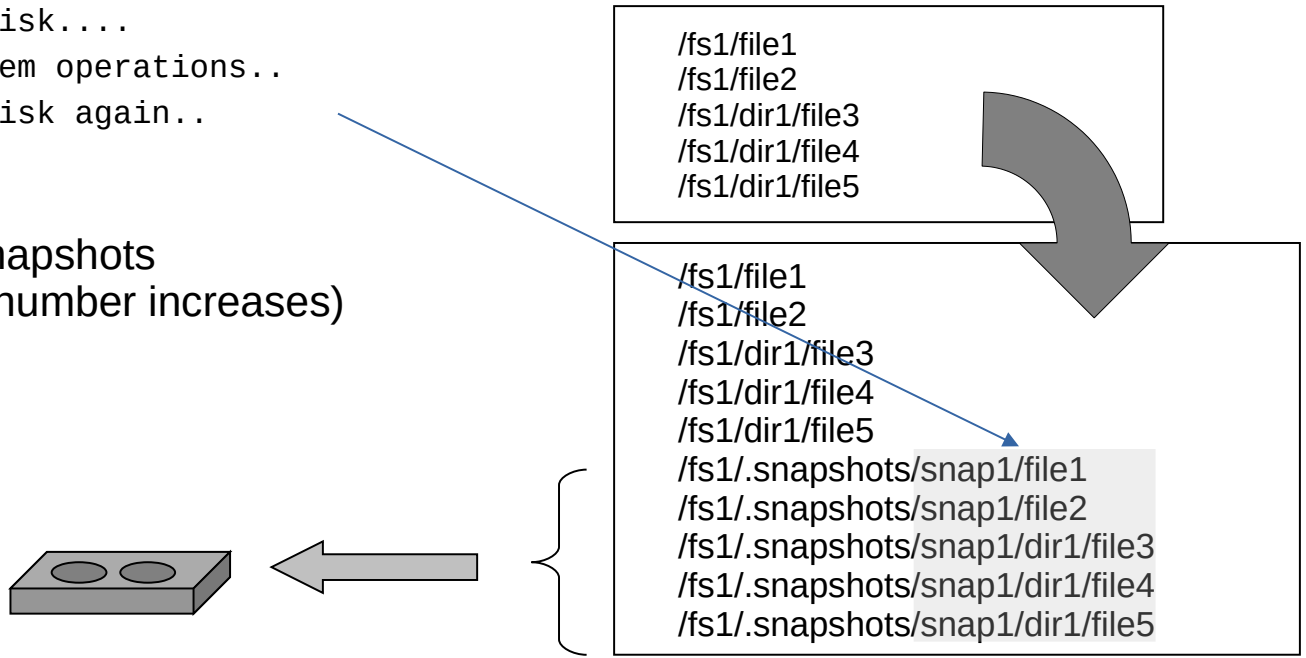
```
                                  Block Limits                    |                File Limits
Name   fileset   type  KB      quota  limit in_doubt  grace   | files   quota limit in_doubt  grace
root   proj1     USR    0          0      0        0   none   |     1       0     0        0   none
root   proj1     GRP    0          0      0        0   none   |     1       0     0        0   none
admin  proj1     GRP    0  268435456      0        0   none   |     0  307200     0        0   none
```

GPFS snapshot

- Creating a snapshot

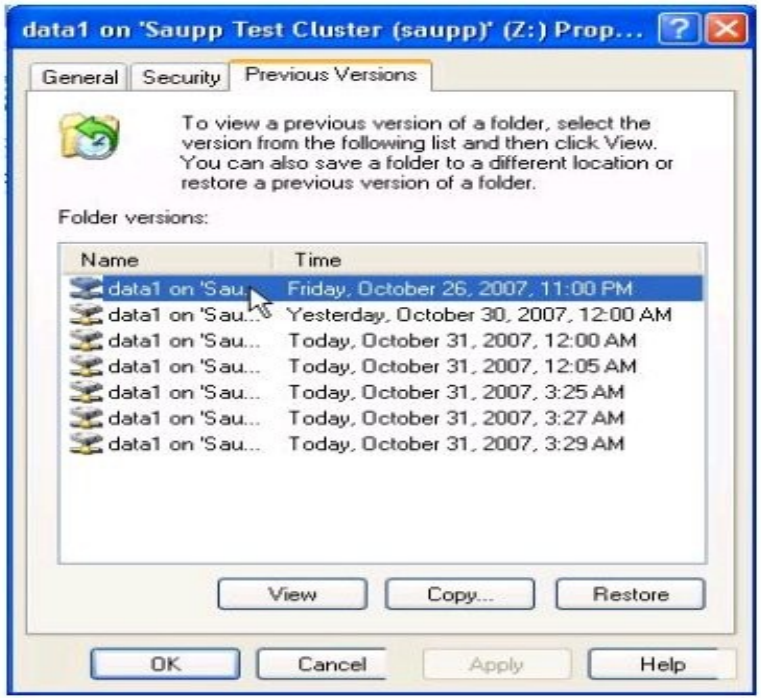
```
# mmcrsnapshot fs1 snap1
writing dirty data to disk....
quiescing all file system operations..
writing dirty data to disk again..
creating snapshot..
resuming operations...
```

- Up to 256 outstanding snapshots (performance impact as number increases)



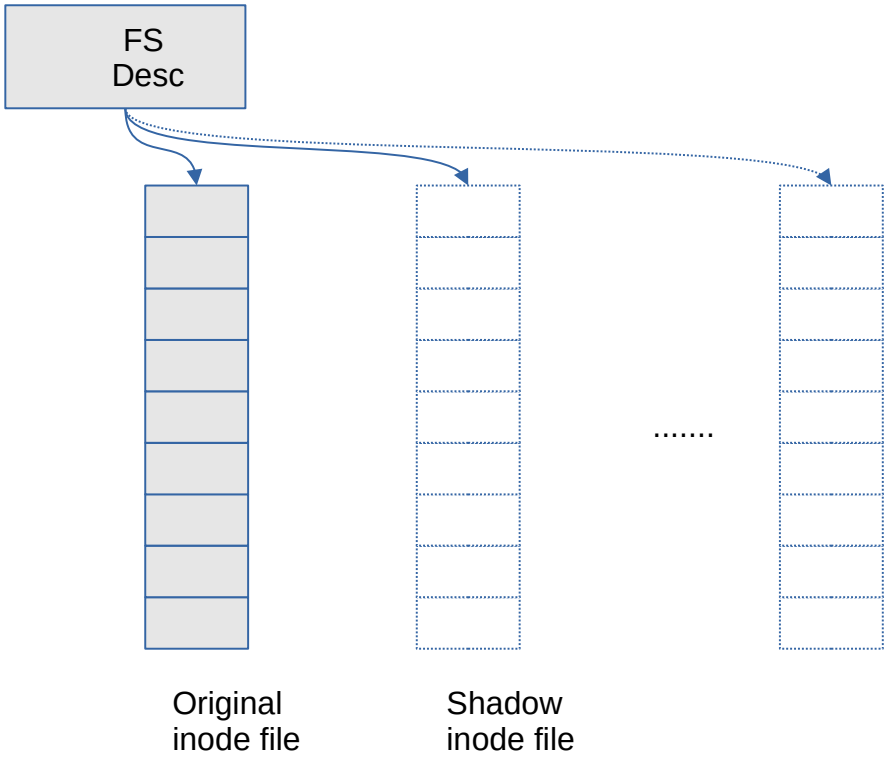
Read only copy, only changes to original file use disk space

Snapshot functionality with windows



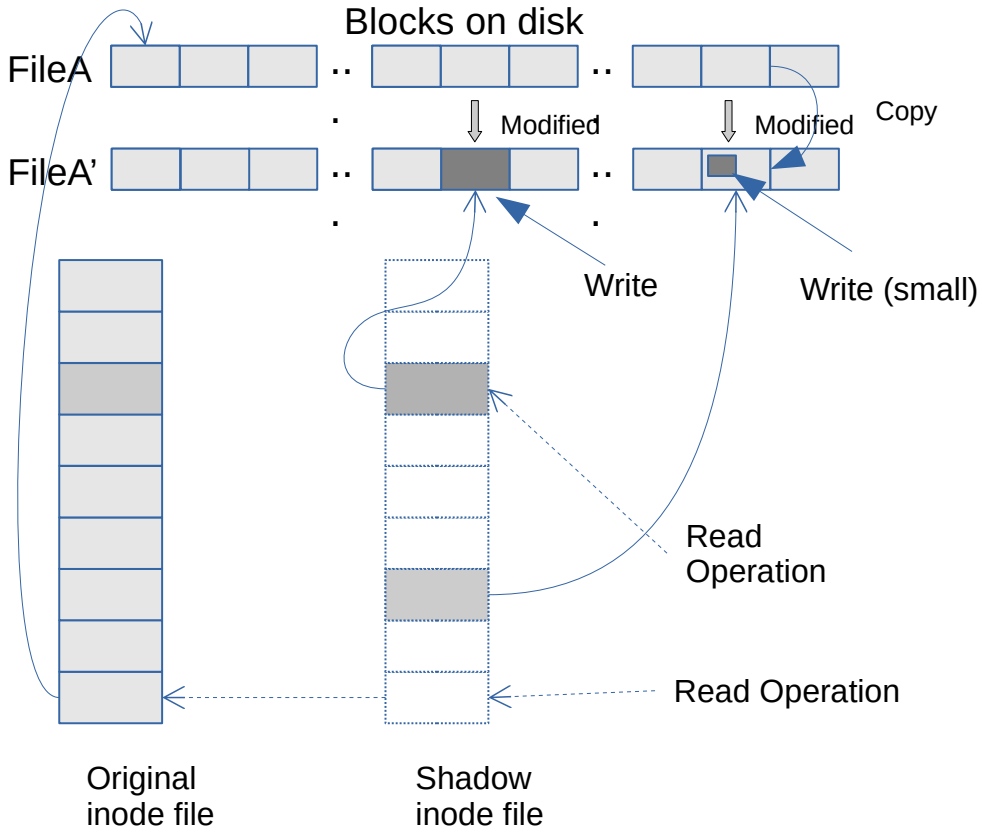
- Snapshot is integration into windows using Volume Shadow Copy Service

Snapshot operation



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

Snapshot operation (cont)

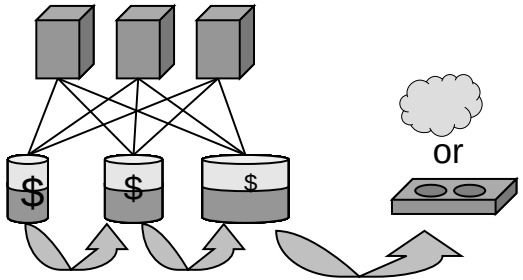


- Snapshots
  - COWOWYHTOROW*
  - Can restore from them
  - Can snapshot filesets (independent)
  - Integrated with mmbackup (only works with Spectrum Protect)
- Operation
  - For most data operations new snapshot data in GPFS is directed into new data blocks and pointers are changed for the version 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"

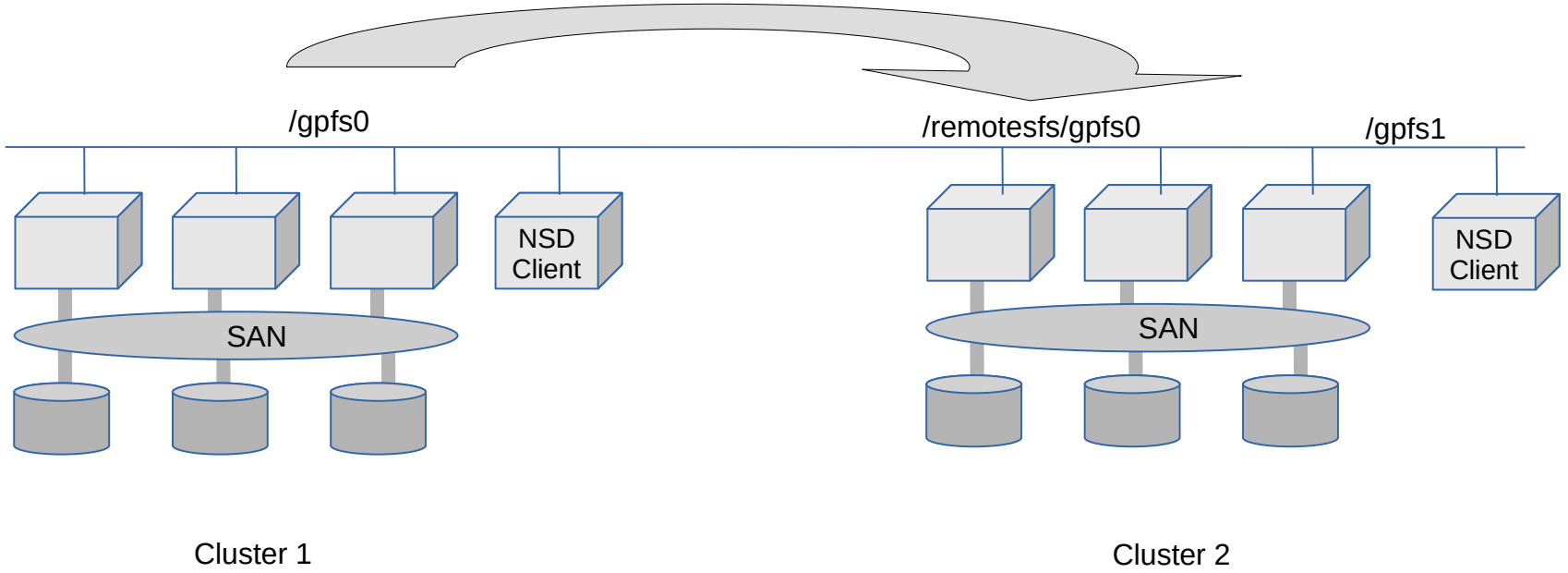
GPFS DMAPI / HSM

- The Open Group has defined a standard API that allows to create extensions to existing file systems.
- 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 / Cloud object store
  - Files can be migrated to tape storage pool, leaving a stub in the file system
  - When stub is accessed, a recall is issued.
  - GPFS does the scanning
    - Example: Rules when file ages, size of file, file system 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



Monitoring, administration and tools

GPFS monitoring

- Global health

```
# mmgetstate -aLs

 Node number  Node name       Quorum  Nodes up  Total nodes  GPFS state   Remarks
------------------------------------------------------------------------------------
       1      gpfs1_priv         2        3          3       active       quorum node
       2      gpfs2_priv         2        3          3       active       quorum node
       3      gpfs3_priv         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 platinum
disk                disk size  failure holds    holds           free in KB          free in KB
name                    in KB    group metadata data        in full blocks        in fragments
--------------- ------------- -------- -------- ----- -------------------- -------------------
Disks in storage pool: platinum (Maximum disk size allowed is 441.49 GB)
nsd3                 52428800        1 no       yes        52355072 (100%)          8056 ( 0%)
nsd4                 52428800       10 no       yes        52355072 (100%)          8056 ( 0%)
                -------------                         -------------------- -------------------
(pool total)        104857600                             104710144 (100%)         16112 ( 0%)
```

mmpmon

- Different modes

email me for example scripts.

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

```
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
.....
```

```
ver
fs_io_s
```

```
_fs_io_s_ _n_ 172.16.1.11 _nn_ ts1 _rc_ 0 _t_ 1248761263
 _tu_ 380682  _cl_ test1.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
...
```

Other monitoring

- Spectrum Scale GUI
  - Monitoring (pmcollectors and pmsensors)
  - Easy to integrate with Grafana / time series database, for example influxDB
- Ganglia, Nagios,...
- Nigel's tool "njmon" also collects GPFS metrics now

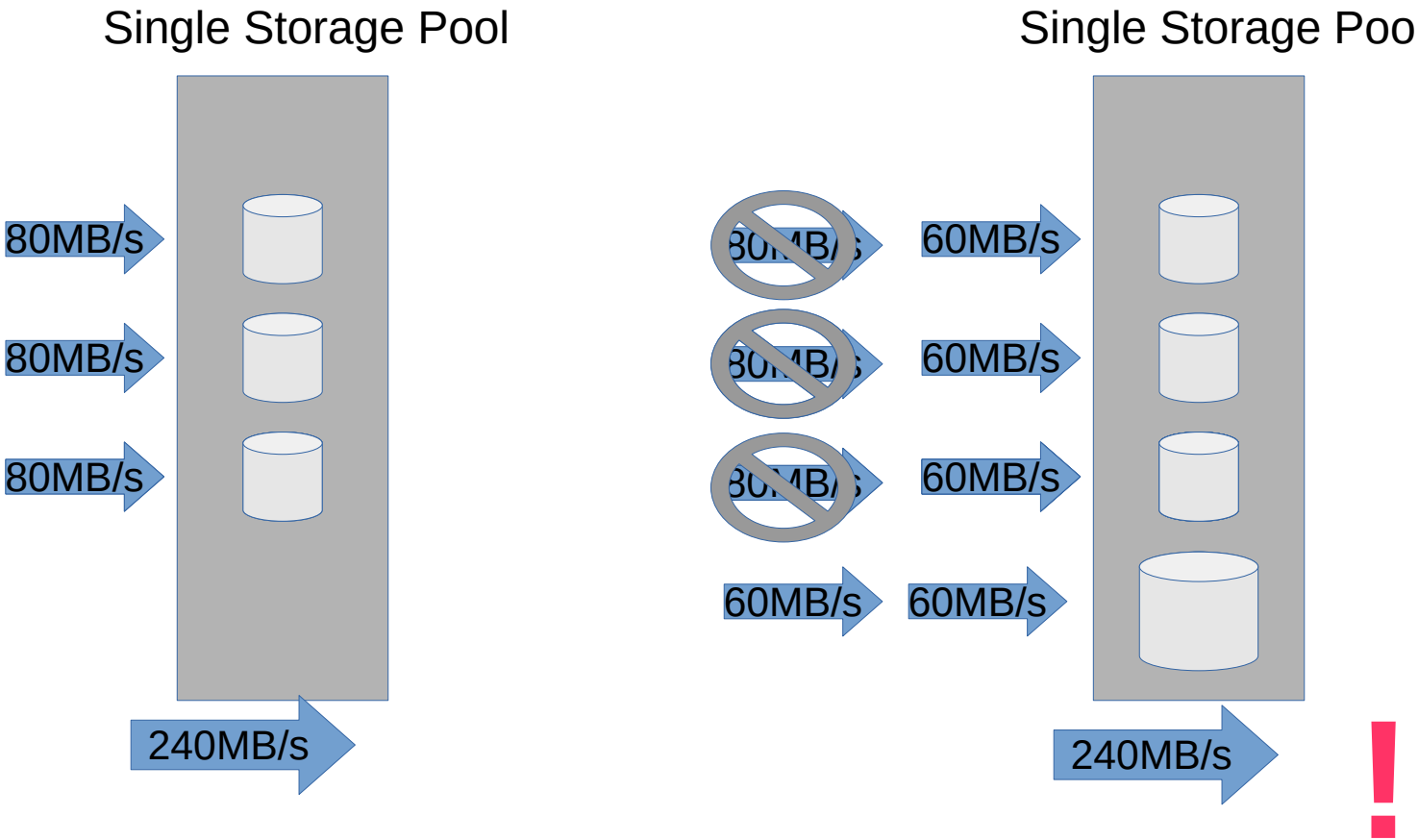
http://nmon.sourceforge.net/pmwiki.php?n=Site.Njmon

Administration

- 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).
- Upgrades
  - Spectrum Scale will support rolling upgrades
  - Check that it is supported to go from your current to your desired version (See GPFS FAQ), some upgrades may require 2 steps.
  - Can keep subset of the nodes running and providing resources.
  - Don’t forget to update file system at end of process
    - mmchcluster -p LATEST
- GPFS GUI
  - Cluster and Storage administration
- Stop a file system automatically mounting on a particular node
  - Create /var/mmfs/etc/ignoreStartupMount.<fs_name>

https://www.ibm.com/docs/en/spectrum-scale?topic=STXKQY/gpfsclustersfaq.html

Management tasks – adding new disks



GPFS tools

- 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 Availability

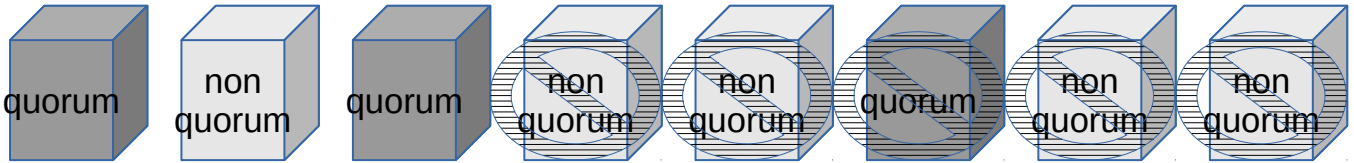
GPFS features and functions - availability

- High availability
  - 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, for exampe:
      - Manual operator intervention
      - Tie breaker
      - Quorum concept: (n/2)+1
  - Same is true for mirrored disks: A quorum of disks guarantees the integrity of file system metadata
  - High Availability and Disaster Resilience
    - It‘s NOT that easy and simple, the devil is in the detail
- Configuration options
  - Quorum
  - File system quorum
  - Failure Groups
  - Replication
  - HA and DR

Quorum

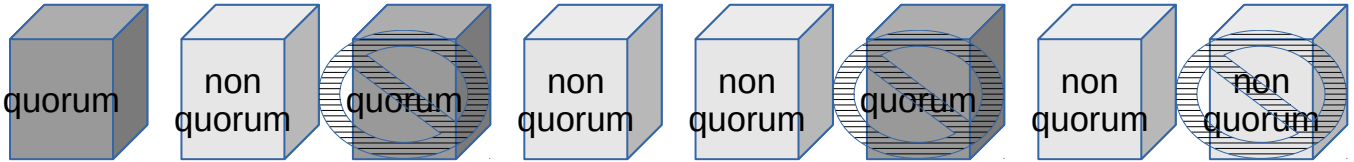
- GPFS quorum = ½ (number of quorum nodes) + 1
- Available since GPFS version 2.2
- Usually the most reliable nodes
- odd number recommended

Quorum = 3, Failed nodes = 5



Cluster still up

Quorum = 3, Failed nodes = 3



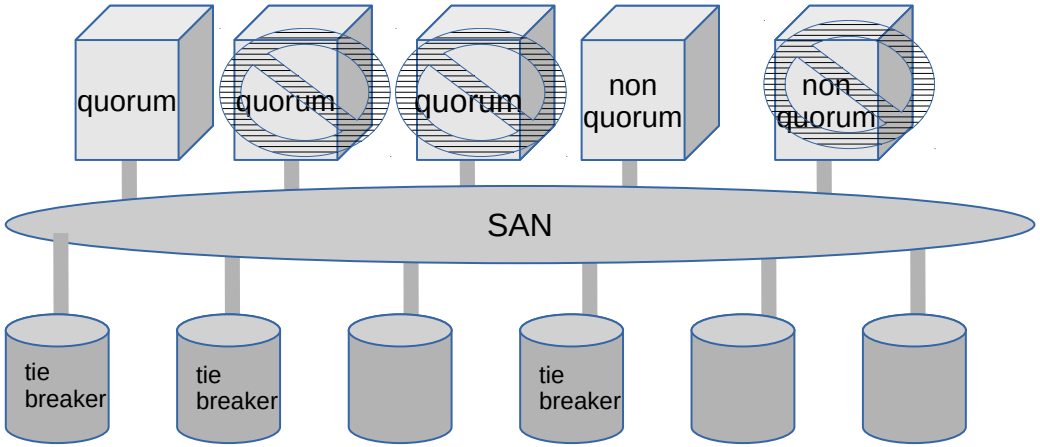
Cluster down

Quorum (cont)

- Managing quorum nodes
  - At creation time either specify quorum 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 servers are allowed
  - Tiebreaker disks need to switched on using

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

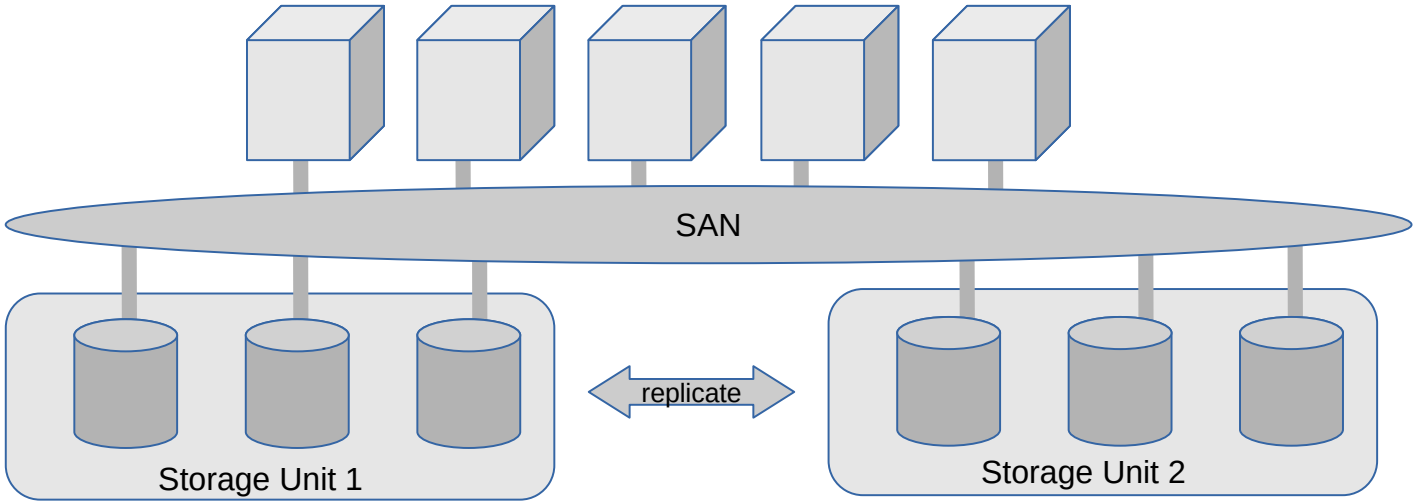
    mmchconfig tiebreakerDisk=no

File system quorum

- Node quorum determines if the cluster will remain active
- File system quorum determines if the file system 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

- Failure groups are a concept to control replication of files
- 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
- After a change, a `mmrestripefs` must then be performed to move data to correct location

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

Failure groups (cont)

- Example of system with two failure groups

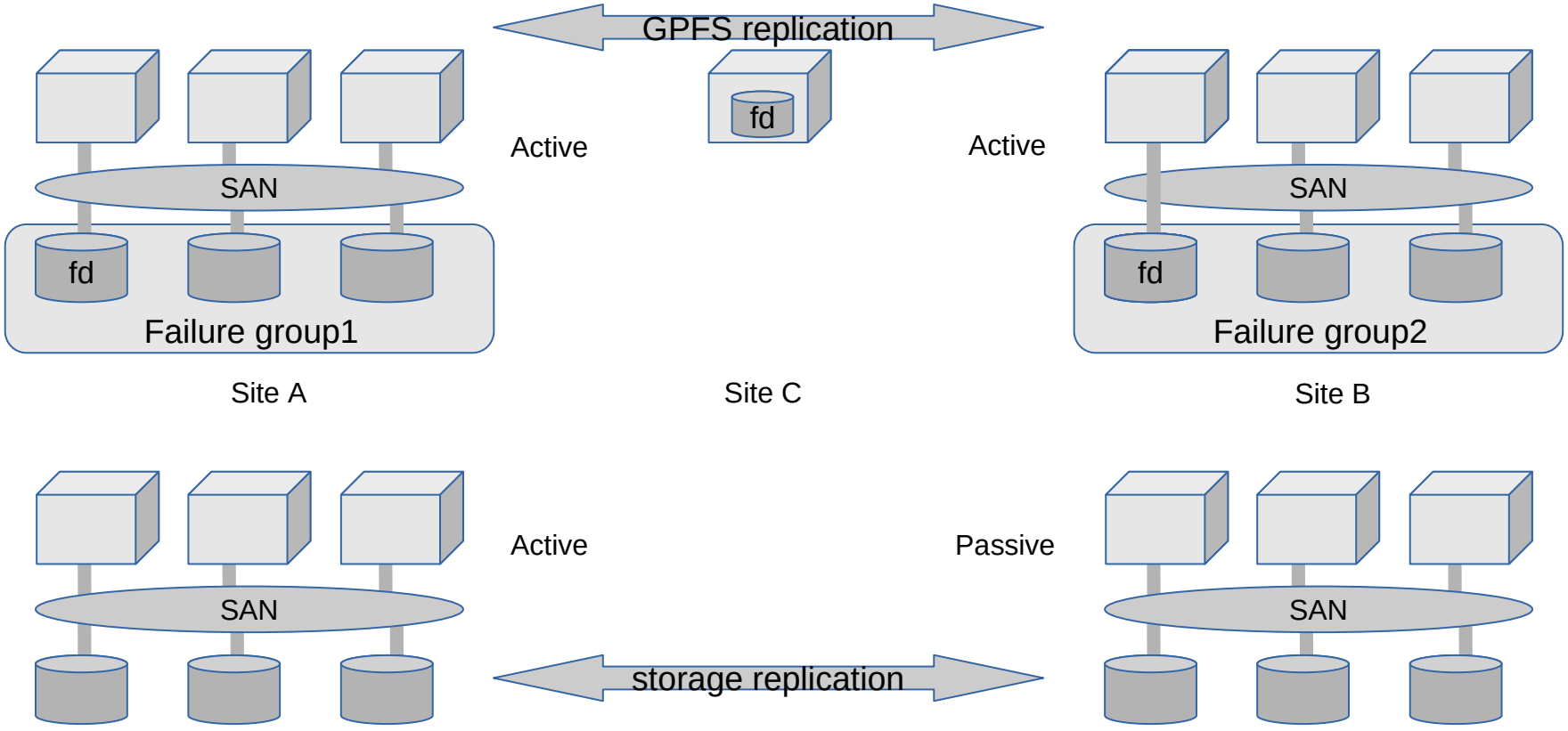
```
# mmlsdisk gpfs0
disk         driver   sector failure holds    holds                            storage
name         type       size   group metadata data  status        availability pool
------------ -------- ------ ------- -------- ----- ------------- ------------ ------------
nsd01        nsd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

Replication

- 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 file system was created and cannot be changed afterwards
  - *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

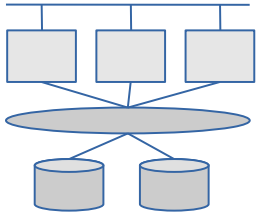


Active File Management and AFM/DR

GPFS Active File Management

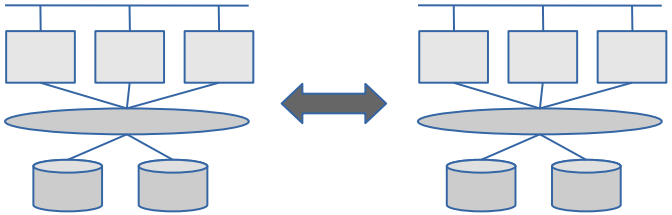
- When GPFS was introduced in 1998 it represented a revolution in file storage. For the first time a group of servers could share high performance access to a common set of data over a SAN or network. The ability to share high performance access to file data across nodes was the introduction of the global 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.

What is Active File Management (was Panache)



1993

GPFS introduced concurrent file system access from multiple nodes.



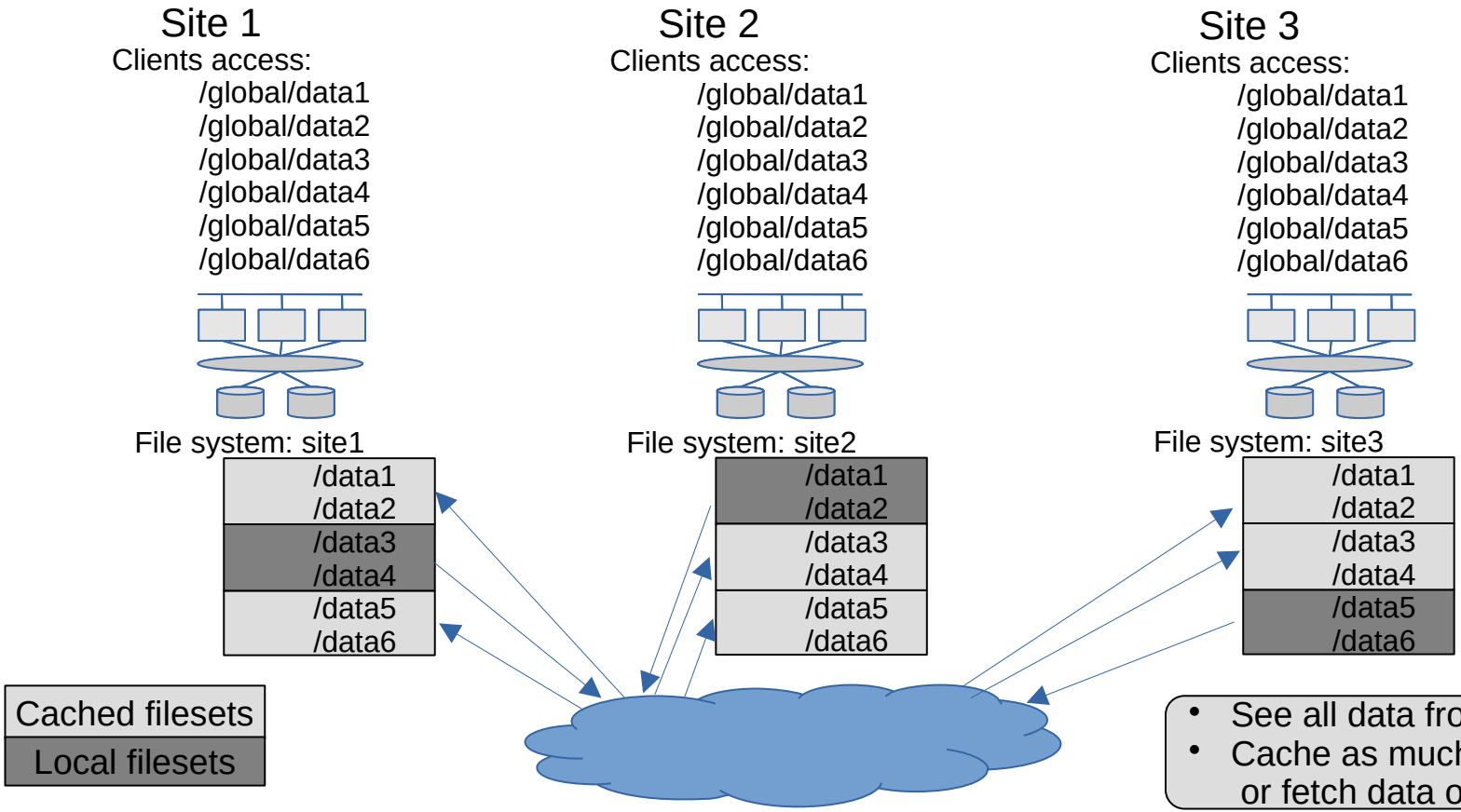
2005

Multi-cluster expands the global namespace by connecting multiple sites

2012

- 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

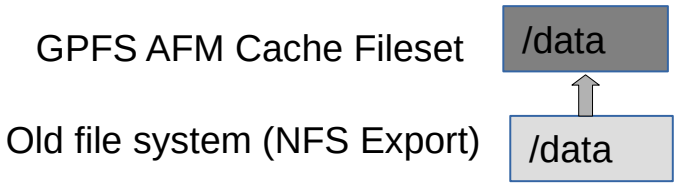
Global namespace with AFM Cache



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

Migrate data into GPFS using AFM

- 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 and copied into the cache fileset on demand or using prefetch. Active users can be migrated over before all of the data is prefetched.



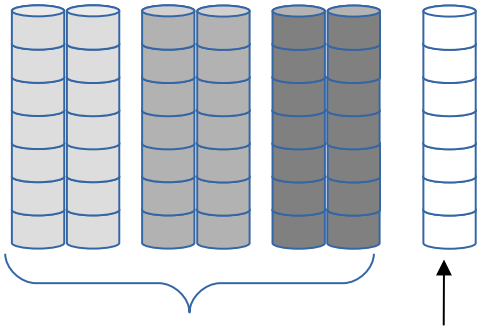
GPFS Native Raid

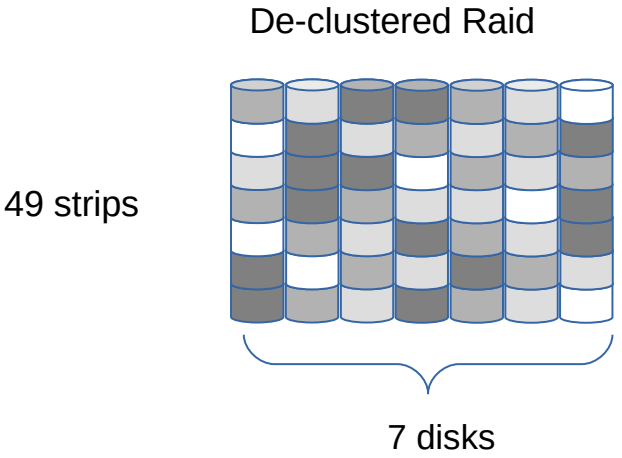
Why GPFS Native Raid

- When building a cluster with multiple pods – slow rebuild of one pod, will affect performance of not only that pod but whole file system (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 it achieves:
  - 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, as the disk can tell you that there is an error) - for silent errors, the disk 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.
  - IBM Labs (Almaden) estimate with just 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)
  - Attach a checksum to the data allows checking of data, but this will not check dropped writes, as old data will match the checksum. To protect against this, the checksum is put at a different location with a version number
  - It is a RAID controller – so need to as well as rebuild, re-balance, scrub and control the scheduling of these operations (setting rate on criticality of the rebuild etc)

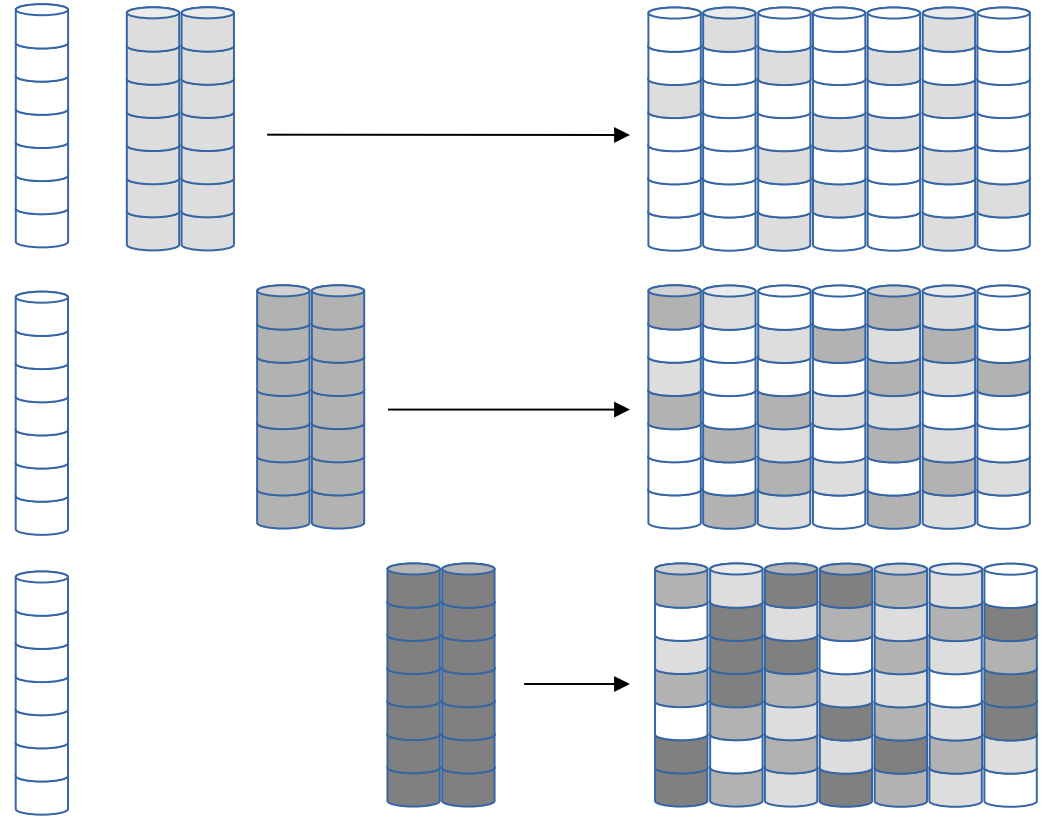
De-clustered RAID example

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



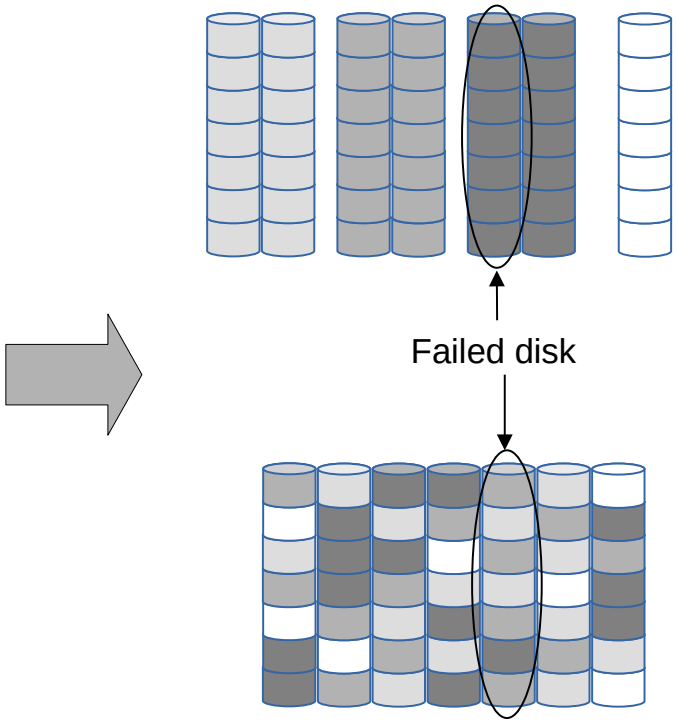


De-clustered RAID example (cont)



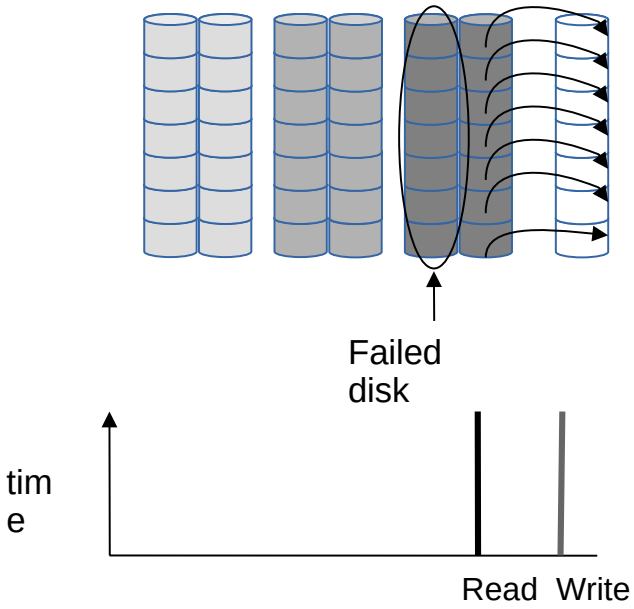
spare disk

3 arrays on 6 disks

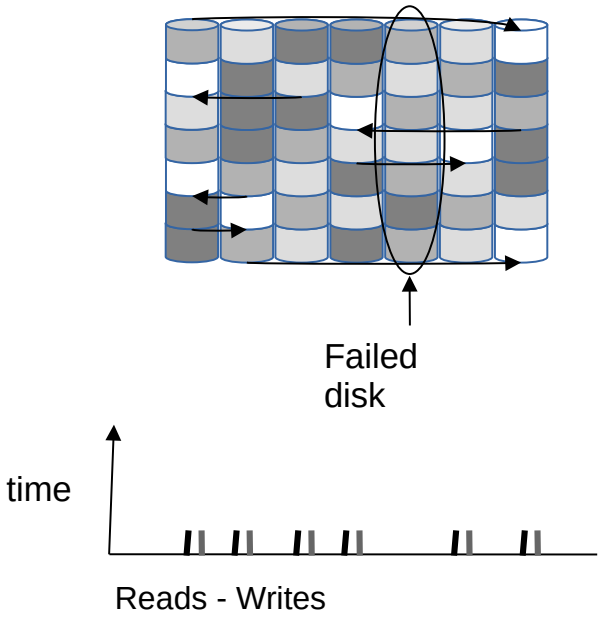


Failed disk

De-clustered RAID example (cont)



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



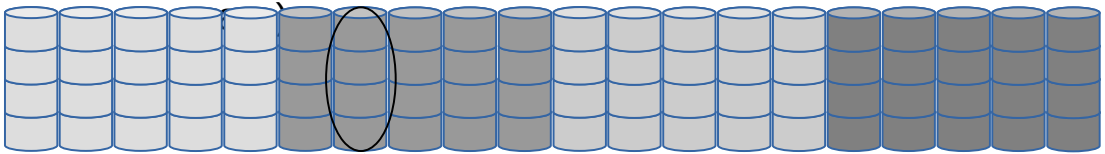
Rebuild activity spread across many disks,
less disruption to user programmes

De-clustering – parallelism applied to spinning disks

- 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

4x4 RAID stripes (data plus parity)



Failed disk

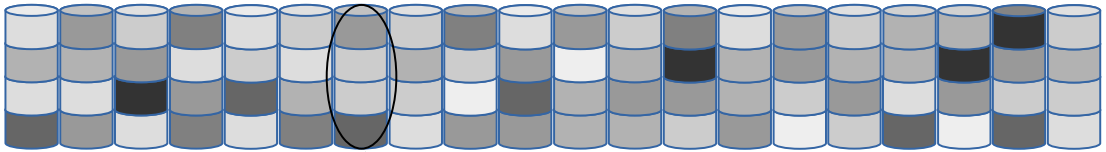
Striping across all arrays, all file accesses are throttled by array 2's rebuild overhead.

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)

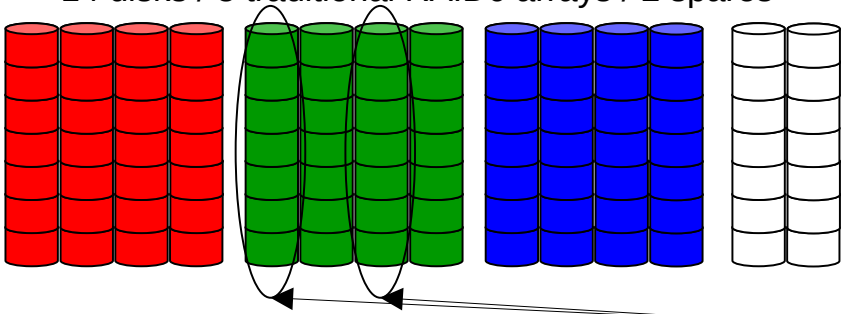


Failed disk

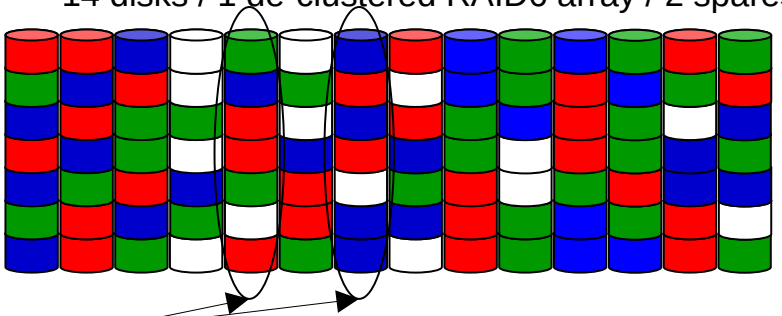
Load on files accesses are reduced by 4.8x(=19/4) during array rebuild.

De-clustered RAID 6 example

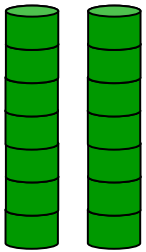
14 disks / 3 traditional RAID6 arrays / 2 spares



14 disks / 1 de-clustered RAID6 array / 2 spares

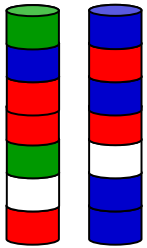


Failed disks



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

IBM Developer Spectrum Scale developer edition

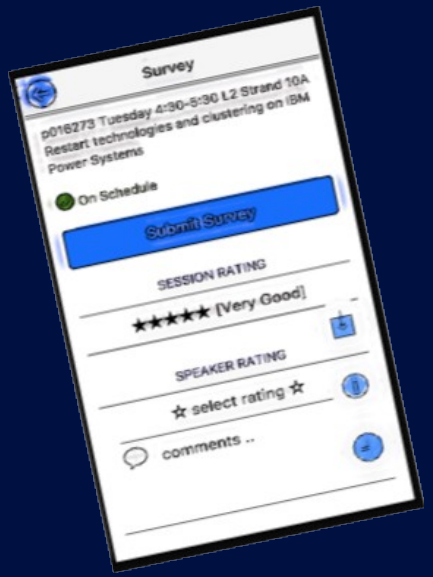
- Learn, develop and test with IBM Spectrum Scale, for non-production use. Create a single global namespace of high-performance or tiered data using new or current storage up to 12TB with no time limits.
  - Derived from DME, all functionality available
    - Limited to 12 TBs: enough for a small test cluster
    - Available from the Scale Marketplace page on ibm.com
    - Initially RHEL x86 only
  - Free for non-production use, e.g. test, learning, upgrade prep...
  - Not formally supported
  - https://www.ibm.com/account/reg/signup?formid=urx-41728
- Join community
  - https://community.ibm.com/community/user/storage/home?_ga=2.219556762.2462946.1629968748-2029247334.1623202591

References

- IBM file and object storage community
  - https://community.ibm.com/community/user/storage/communities/community-home?CommunityKey=1142f81e-95e4-4381-95d0-7977f20d53fa
- IBM Spectrum Scale documentation – FAQ
  - https://www.ibm.com/docs/en/spectrum-scale?topic=STXKQY/gpfsclustersfaq.html
- IBM Spectrum Scale (GPFS) Global User Group
  - https://community.ibm.com/community/user/integration/communities/globalgrouphome?CommunityKey=64bee33a-49c0-4158-9ce7-6012673e77e3
- Spectrum Scale User Group: Digital
  - https://www.spectrumscaleug.org/

Thank you
s203870 - Introduction to Spectrum Scale

Antony (Red) Steel
antony.steel@belisama.com.sg



Please don’t forget to complete the session evaluation!

Notices and disclaimers

U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those

customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers

- Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**
- The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.
- IBM, the IBM logo, ibm.com and [names of other referenced IBM products and services used in the presentation] are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml

Backup slides

Some changes in version 5.x

- 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 file system 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
  - Providers flexible API which allows programmatic actions to be taken based on file system events. Can be run against directories, filesets, and inode spaces.
- Deployment toolkit
  - Designed to simplify GPFS deployments now has support for System Z and Ubuntu
  - Improvements in the toolkit for different upgrade scenarios
- 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.

Some further changes in version 5.x (cont)

- 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 file system support, tier different fileset to different cloud containers, enhanced support for multiple cloud accounts and containers.
- GPFS Raid
  - Greater throughput; faster rebuild times; end to end checksum
- GPFS File Placement Optimisation
  - GPFS Shared nothing clusters

Some further changes in version 5.x (cont)

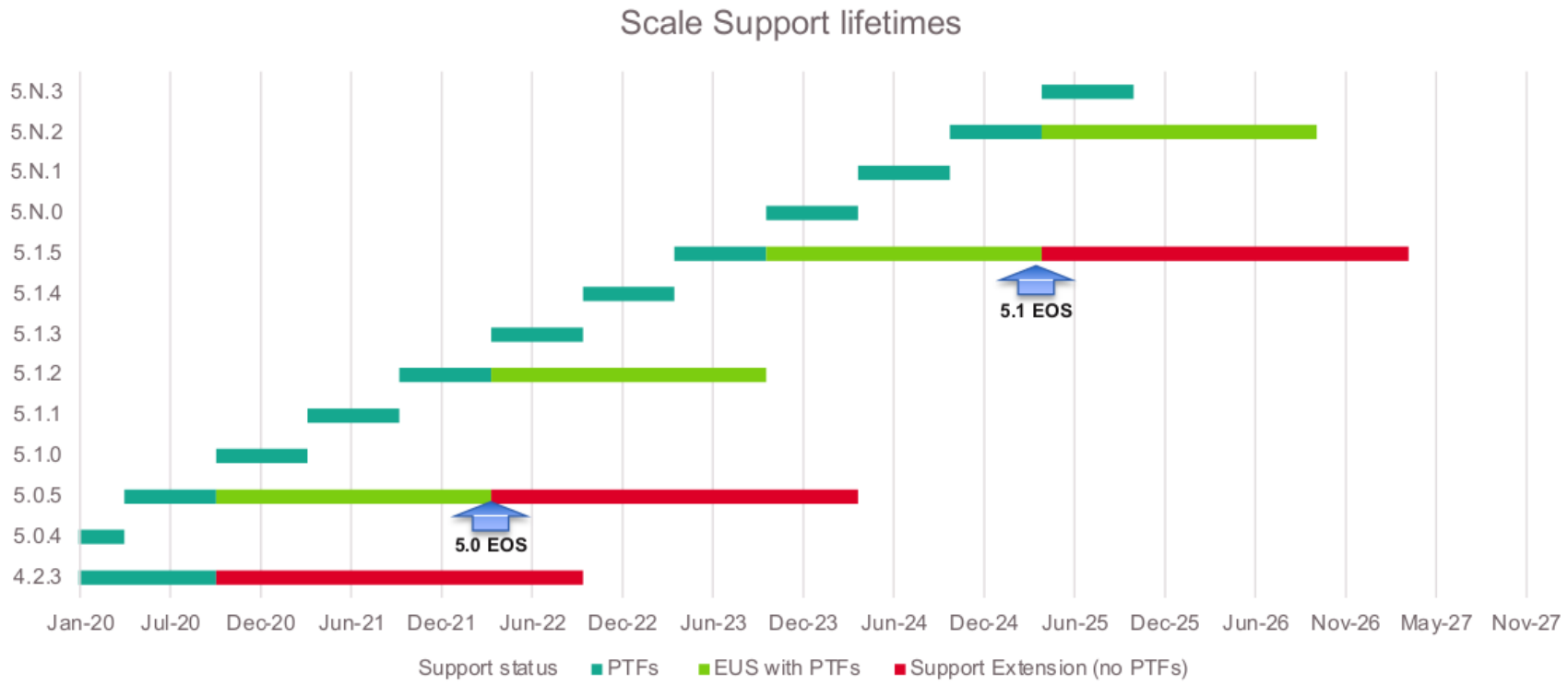
- High Performance Extended Attributes
  - GPFS has long supported the use of extended attributes, though in the past they were not commonly used, in part because of performance concerns. In GPFS 3.4, a comprehensive redesign of the extended attributes support infrastructure was implemented, resulting 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.
- AFM improvements
  - 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

Futures and roadmap

Spectrum Scale roadmap and release cadence

- New Scale version or release every three years (min)
  - Mod packs roughly every 6 months
- Extended Update release every 18 months
  - every third Mod pack comes with Extended Update Support (EUS)
  - PTF support on Extended update release and on Mod packs (ie 18 or 6 month support)
  - 5.0.5 is the first Extended Update release

Release planning



Linux release planning

Spectrum Scale EUS and RHEL EUS alignment

Scale release	Planned date	7.7*	7.9	8.1	8.2	8.4	8.6	8.8	9.x?
5.0.4	*4Q2019*	■		■					
5.0.5 EUS	**2Q2020**	■	■	■	■	■			
5.1.0	*4Q2020*	■	■	■	■				
5.1.1	*2Q2021*			■	■				
5.1.2 EUS	**4Q2021**				■	■	■		
5.1.3	*2Q2022*				■	■			
5.1.4	*4Q2022*					■	■		
5.1.5 EUS	**2Q2023**						■	■	■
5.N.0	*4Q2023*						■	■	
5.N.1	*2Q2024*							■	■

*See Knowledge Center for details of RHEL7 support. Subject to changes to Red Hat release plans

Spectrum Scale EUS and SLES 12&15 alignment (SLES packs 12-18 months)

Scale release	Planned date	12 SP4 (12/2018)	*12 SP5* (6/2020)	*12 SP6* (6/2021?, fin)		15 SP1 (6/2019)	*15 SP2* (12/2020)	*15 SP3* (6/2022)	*15 SP4* (12/2023)
5.0.4	*4Q2019*	■				■			
5.0.5 EUS	**2Q2020**	■	■	■		■	■		
5.1.0	*4Q2020*					■	■		
5.1.1	*2Q2021*					■	■		
5.1.2 EUS	**4Q2021**						■	■	
5.1.3	*2Q2022*						■	■	
5.1.4	*4Q2022*						■	■	
5.1.5 EUS	**2Q2023**							■	■
5.N.0	*4Q2023*								■
5.N.1	*2Q2024*								■

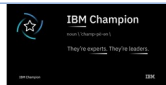
Spectrum Scale EUS and Ubuntu LTS alignment

Scale release	Planned date	18.04 (4/18 – 10/20)	20.04 (4/20 – 10/22)	22.04 (4/22 – 10/24)	24.04 (4/24 – 10/26)
5.0.4	*4Q2019*	■			
5.0.5 EUS	**2Q2020**	■	■		
5.1.0	*4Q2020*		■		
5.1.1	*2Q2021*		■		
5.1.2 EUS	**4Q2021**		■	■	
5.1.3	*2Q2022*		■	■	
5.1.4	*4Q2022*			■	
5.1.5 EUS	**2Q2023**			■	■
5.N.0	*4Q2023*			■	
5.N.1	*2Q2024*			■	■

Spectrum Scale futures

- Strategic workloads
  - AI & ML (including in research and academia)
  - Big data analytics (Finance, Life Science, Genomics, ADAS)
  - Modeling and simulation (HPC)
  - Data lakes for analytics or archive
- Performance leadership
  - Exploit NVMe and NVMeoF more efficiently
  - Maximise GPU performance (Nvidia)
  - Optimise for Storage Class Memory
- Increase full data lifecycle leadership
  - High Performance Storage Tier (SCM, NVMe)
  - Support for the AI pipeline and for high performance analytics (including HPT data migration)
  - Improve support for data reduction technologies
- Improve “fit to enterprise”
  - Improved deployment and installation, including integration with data centre automation tools
  - Support for enterprise-preferred monitoring and management frameworks
  - Lead in enterprise requirements including security, RBAC, audit, minimisation of privileges required to operate Scale
- Container and Cloud deployment
  - Support OpenShift workloads and other containerised environments
  - Provide tight integration (“in the cloud”) with leading public clouds starting with IBM Cloud
  - Extend hybrid cloud support to optimise for additional workloads including archive, DR, bursting

Belisama



Introduction to problem determination

- Initial investigation
  - Is the problem on one node, several, all?
  - Is the problem with the daemon, some file systems, all?
  - Disk / Storage / Storage network problems
  - Network problems
  - Are other commands having problems (communications between nodes
  - What is in the GPFS logs (Warning these logs roll with each restart of the gpfs daemon)
    - /var/adm/ras/mmfs.log.latest
      /var/adm/ras/mmfs.log.previous
      /var/adm/ras/mmfs.log.* (older logs?)
    - Daemon started correctly
    - Daemon connected to the cluster
    - Cluster progressed correctly – any errors

```
Mon Jan 16 15:26:29 2006: mmfsd initializing. {Version: Development Built: Jan 16 2008
12:47:11} ...
Mon Jan 16 15:26:29 2006: Connecting to 192.168.1.2 node2
Mon Jan 16 15:26:31 2006: Accepted and connected to 192.168.1.2 node2
                          mmfsd will not be ready until quorum is reached.
Mon Jan 16 15:26:38 2006: mmfsd ready
Mon Jan 16 15:26:38 EST 2006: mmcommon mmfsup invoked
                          nonquorum nodes might join at any time before or after mmfsd ready.
Mon Jan 16 15:26:53 2006: Accepted and connected to 192.168.1.8 node8
```

Introduction to PD (cont)

- Initial investigation (cont)
  - Anything interesting in the system logs?
    - /var/log/messages | system error report
    - Did node(s) reboot?
      - Anything interesting in the boot log?
  - (Linux:) Is it related to the GPL layer?
    - Kernel change?
    - See /usr/lpp/mmfs/src/README
  - Do the remote commands/copy command work?
  - gpfs.snap collects useful data (from all nodes, one, working collective..)
    - Good general starting point, but not everything (eg mmlsnsd -M for specific problems)
    - May hang under some conditions if a deadlock

Introduction to PD (cont)

```
# mmlscluster
GPFS cluster information
========================
  GPFS cluster name:         422.test.red.com
  GPFS cluster id:           12398410922139748073
  GPFS UID domain:           test.red.com
  Remote shell command:      /usr/bin/ssh
  Remote file copy command:  /usr/bin/scp

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

 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
   3   ts3.red.com                 172.16.1.13      ts3.red.com           quorum-manager
```

Note which network interfaces are in use by GPFS.

Introduction to PD (cont)

- What is the role of each of the nodes?
  - Did a file system have an issue – find it's manager?
  - Did a node have an issue, what was it's role?

```
mmlsmgr

file system      manager node        [from 192.168.1.5 (node1]
---------------- ------------------
gpfs1            192.168.1.5 (node1)
tst1             192.168.1.5 (node1)
gpfs2            192.168.1.2 (node2)
tst2             192.168.1.2 (node2)
```

Introduction to PD (cont)

- Gathering data
  - Collect a snap from some or all nodes
    - gpfs.snap
    - Collects general data – and may not work on some nodes with problems
    - If you have an idea of the problem, collect more data related to it
  - mmfsadm dump (mmdiag is now the preferred command)
    - Service aid to dump the internal states of the daemon.
    - mmfsadm dump all will capture the maximum amount of data.
    - However, “dump all” may conflict with a daemon that is doing real work at the time.
    - Beware that it might segfault in that case.
  - mmfsadm dump waiters (shows threads waiting for resources or responses) is (relatively!) safe.
  - mmtrace
    - Service aid to trace GPFS execution on a node.
    - Used to capture sequence of events leading to a problem
    - mmtrace -help shows options and explanation.
    - Trace file size is fixed, and wraps around when full.
      - Be aware of environment TRCFILESIZE to increase trace file size.
    - mmchconfig trace='traceclass level traceclass level' -i
      - Set trace levels now and at restart (whole cluster.)
    - mmfsadm trace traceclass level traceclass level
      - Set trace levels now (local node.)
    - mmfsadm showtrace

Introduction to PD (cont)

- Check for non GPFS problems
  - At least one NSD Server available for each NSD / No SAN connectivity issues
  - When not using CCR, no secondary data configuration server defined and the primary is not available
  - Remote command and copy not set up on all nodes
  - Network problem (see timeouts in the gpfs logs)
  - Replication not configured and storage problem
  - Follow local PD for storage
- mmfsadm (now can use mmdiag)
- Subsections of mmfsadm dump all, also available separately:
  - mmfsadm dump config - Showing the Daemon Configuration
  - mmfsadm dump tscomm - Showing Node Connection
  - mmfsadm dump cfgmgr - Showing Config Manager Information
- mmfsadm dump config
  - Easy way to check daemon configuration settings that are actually in effect (as opposed to what is stated by mmlsconfig.
    - For example: user might have used mmchconfig to modify some settings, but not specified the -i option (for immediate change).
    - User might have used mmchconfig to modify some settings for which immediate change (-i option) is not available.
  - Very long list of interesting data
- mmdiag (now recommended)
  - Much safer and friendlier way to collect PD data (mmfsadm has a tendency to halt a system – particularly dump all and dump threads)
    - mmdiag [--help --all --version --waiters --threads --memory --network --config --trace --iohist --tokenmgr --stats]

Introduction to PD (cont)

- Look at the pagepool on all nodes:
  - mmdsh mmfsadm dump config | grep pagepool
    node1: pagepool 536870912
    node2: pagepool 536870912
    node3: pagepool 536870912
    node4: pagepool 536870912
  - Sometimes the OS will not allow the full amount of the pagepool, so what is listed in mmlsconfig might be more than the node actually has

Introduction to PD (cont)

- mmfsadm dump tscomm
  - Lots of great stuff, but let's focus on the highlights:

```
Pending messages:
  msg_id 16075, service 16.1, msg_type 1 'nsdMsgWrite', n_dest 1, n_pending 1
  this 0x86A96A0, n_xhold 1, ccP 0x93929548 cbFn 0x0
  sent by 'File block write fetch handler' (0xA7B12708)
  dest 192.168.1.5          status pending     , err 0, reply len 0
```

Where:

msg_id: a unique number assigned to that RPC.

'quoted name': the RPC name, usually descriptive.

n_dest: number of destination nodes for the RPC.

n_pending: number of destination nodes which have yet to reply to this message.

sent by: name of thread sending the RPC. Name is descriptive.

dest: for each destination, show destination node, status, and other info.

  - And:

```
Waiting messages (none)
```

  - This is good – if there are waiting messages, means that GPFS didn't have enough worker threads to process incoming messages

Introduction to PD (cont)

- mfsadm dump cfgmgr
  - Some interesting fields:

```
NClusters 1   <=== number of clusters, Cluster data follows
Cluster Configuration [0] "parzival.localdomain": Type: 'LC' id C0A8010445E49F16
No of nodes: 3 total, 3 local, 3 core nodes.  <== node counts
Cluster configuration manager is 192.168.1.5 (other node)
```

  - On non-config mgr, disk lease information for the node:

```
lastFailedLeaseGranted: 4308242.83 (26 seconds ago)
lastLeaseObtained       4308267.14, 33.11 sec left (ok)
lastLeaseReplyReceived  4308267.14 = 0.00 sec after request
```

Introduction to PD (cont)

- mmfsadm dump cfgmgr
  - Connectivity (from non-config mgr, subnets)

```
node                    primary      admin  --status---   join fail SGs other ip addrs,
     no   idx host name       ip address     func tr p    rpc seqNo cnt mngd last failure
     ---- ----- --------------   ------------ -----  ----------- ------ ---- ----
     ------------------
     3      2 voyager        192.168.1.2     --l -- J     up      4    1   0 192.168.2.2
                                                                          2007-02-27 19:09:07
     2      0 parzival       192.168.1.4     -ml -- J     up      1    0   0 192.168.2.4
     1      1 yankeeclipper  192.168.1.5     q-l -- J     up      1    0   0 192.168.2.5
```

    - primary ip address: never the subnet
    - admin func:
      l==local node m==manager q==quorum n== new node Q==new quorum node
      tr (transitions): --==normal, jw==join-wait, lw==leave-wait
      status: -==none, j==joining, J==joined, l==leaving
    - rpc: up, down, failed join seqNo: larger means joined later
      sgs mngd: always zero at non-config mgr node.
    - Note subnet addresses and timestamp of last failure.

Introduction to PD (cont)

- mmfsadm dump cfgmgr
  - Connectivity (from non-config mgr, subnets)

```
node                    primary      admin  --status---   join fail SGs other ip addrs,
     no   idx host name       ip address     func tr p    rpc seqNo cnt mngd last failure
     ---- ----- --------------   ------------ -----  ----------- ------ ---- ----
     -------------------
     3      2 voyager        192.168.1.2     --l -- J     up      4    1   0 192.168.2.2
                                                                          2007-02-27 19:09:07
     2      0 parzival       192.168.1.4     -ml -- J     up      1    0   0 192.168.2.4
     1      1 yankeeclipper  192.168.1.5     q-l -- J     up      1    0   0 192.168.2.5
```

    - The SG managed count is accurate.
    - Lease timestamps are the last 5 digits of system timestamp in seconds.
    - Meant to examine turnaround time.
    - Lease renewal statistics are included (times in seconds):
    - Summary of lease renewal round-trip times:
      - Number of keys = 3, total count 56
      - Min 0.0 Max 1.0, Most common 0.5 (29)
      - Mean 0.5, Median 0.5, 99th 1.0
    - The config manager version of “dump cfgmgr” shows disk lease data.
    - Config Manager data also shows a list of file systems and their managers.

Introduction to PD (cont)

- Long waiters?
  - Lock / hang?
    - Find out what nodes are managing what file systems
      - mmlsmgr
    - Create a file with the names of all nodes in the cluster
      - /tmp/my_wcoll for example
      - Set environmental variable
      - export WCOLL=/tmp/my_wcoll
    - Collect waiters information
      - mmdsh ‘/usr/lpp/mmfs/bin/mmfsadm dump waiters > /tmp/gfps.all.waiters’

Introduction to PD (cont)

```
 mmfsadm dump waiters

x8484D10 waiting 0.001328000 seconds, NSD I/O Worker: on ThCond 0x8497D78 (0x8497D78)
  (MsgRecord), reason 'waiting for RPC replies' for getData on node 192.168.1.2

....

0x8483150 waiting 0.000547000 seconds, NSD I/O Worker: for I/O completion on disk hda14
0x8482810 waiting 0.000611000 seconds, NSD I/O Worker: on ThCond 0x8497D78 (0x8497D78)
  (MsgRecord), reason 'waiting for RPC replies' for getData on node 192.168.1.2
0x8485650 waiting 0.000967000 seconds, NSD I/O Worker: for I/O completion on disk hda15
0x84843D0 waiting 0.000991000 seconds, NSD I/O Worker: for I/O completion on disk hda15
0x84A0DC0 waiting 0.005947000 seconds, Unused inode prefetch: on ThCond 0x849FD58 (0x849FD58)
  (MsgRecord), reason 'waiting for RPC replies' for NSD I/O completion on node 192.168.1.8
0x849ECD0 waiting 0.013165000 seconds, Unused inode prefetch: on ThCond 0x84A4058 (0x84A4058)
  (MsgRecord), reason 'waiting for RPC replies' for NSD I/O completion on node 192.168.1.8
0x84A8EA8 waiting 0.007731000 seconds, Symlink handler: on ThCond 0x849A7A8 (0x849A7A8)
  (MsgRecord), reason 'waiting for RPC replies' for NSD I/O completion on node 192.168.1.8
0x844F6F8 waiting 0.014779000 seconds, Sync handler: on ThCond 0x849CE48 (0x849CE48)
  (MsgRecord), reason 'waiting for RPC replies' for NSD I/O completion on node 192.168.1.8
```

Introduction to PD (cont)

- Lock / hang (cont)
  - Collect a trace on all nodes
    - mmtracectl --set --trace=def --trace="tm 4 lock 4 lock 4"\
      --aix-trace--buffer-size=400000000\ --trace-file-size=200000000 -N all
    - mmtracectl –start –N all
    - sleep 20
    - mmtracectl –stop –N all
    - mmtracectl –off –N all
  - Collect dump all on the affected nodes
    - Put only the affected nodes into the working collective.
      export WCOLL=/tmp/my_wcoll
      mmdsh '/usr/lpp/mmfs//bin/mmfsadm dump all > /tmp/$(hostame –s)_dumpall'

Note: mmfsadm dump all can cause GPFS to assert in some instances, thus approval must be obtained from the customer first. However in cases of a lock – this data is vital for problem resolution – and intervention will be required anyway to fix the problem.
Thus why we now have mmdiag

Introduction to PD (cont)

- Lock / hang (cont)
  - Collect files for support and recover GPFS
  - Depending on the type of problem, collect the above data, for example:
    - Performance - waiters and trace
    - Lock - waiters, trace and dump
  - Resolution/Recommendation: Create a script on all nodes to collect the data should a GPFS locking problem occur and add to procedure for system administrators

Introduction to PD (cont)

- mmdiag –waiters

1. Look at the time for operations
   - Some operations take a while so look carefully
2. If there is an IP address, follow the trail
   - Log into the node IP in the waiter and see what is happening

```
# mmdiag --waiters
0x100FB8290 waiting 0.009309167 seconds, Unused inode prefetch: on ThCond 0x10104F218 (0x10104F218) (MsgRecord), reason 'RPC wait' for NSD I/O completion
0x100FB8640 waiting 0.017657334 seconds, Msg handler quotaMsgRelinquish: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection
                                         for sending msg'
0x10101DAB0 waiting 0.015502500 seconds, Writebehind worker: on ThCond 0x100F86608 (0x100F86608) (MsgRecord), reason 'RPC wait' for NSD I/O completion
0x101020C30 waiting 0.010179334 seconds, Clean buffer: on ThCond 0x100F89EC8 (0x100F89EC8) (MsgRecord), reason 'RPC wait' for NSD I/O completion
0x101029E90 waiting 0.010610000 seconds, Clean buffer: on ThCond 0x100F891E8 (0x100F891E8) (MsgRecord), reason 'RPC wait' for NSD I/O completion
0x101028090 waiting 0.000507417 seconds, Clean buffer: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection for sending msg'
                                         for NSD I/O completion
0x101026EA0 waiting 0.015334917 seconds, Msg handler quotaMsgRelinquish: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection
                                         for sending msg'
0x101025CB0 waiting 0.008003000 seconds, Msg handler getData: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection for sending msg'
0x100FD3EC0 waiting 0.021359917 seconds, Clean buffer: on ThCond 0x10104F838 (0x10104F838) (MsgRecord), reason 'RPC wait' for NSD I/O completion
0x100FCA490 waiting 0.009490917 seconds, Writebehind worker: on ThCond 0x100F8AB18 (0x100F8AB18) (MsgRecord), reason 'RPC wait' for NSD I/O completion
0x100FD8C60 waiting 0.015392084 seconds, Clean buffer: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection for sending msg'
                                         for NSD I/O completion
0x100F80B60 waiting 0.015440667 seconds, Clean buffer: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection for sending msg'
                                         for NSD I/O completion
0x100F7E200 waiting 0.012739334 seconds, Msg handler quotaMsgUpdateUsage: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection
                                         for sending msg'
0x100F77720 waiting 0.006144500 seconds, Msg handler getData: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection for sending msg'
0x100F7B810 waiting 0.019238000 seconds, Writebehind worker: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection for sending msg'
                                         for NSD I/O completion
0x100F807D0 waiting 0.039823250 seconds, Clean buffer: on ThCond 0x100F8DDF8 (0x100F8DDF8) (MsgRecord), reason 'RPC wait' for NSD I/O completion
0x100F7F5E0 waiting 0.005521334 seconds, Msg handler quotaMsgRelinquish: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection
                                         for sending msg'
0x100F7E3F0 waiting 0.012592584 seconds, Msg handler getData: on ThCond 0x100F79C80 (0x100F79C80) (TcpConn use), reason 'waiting for exclusive use of connection for sending msg'
0x100F1FD10 waiting 0.010031917 seconds, Clean buffer: on ThCond 0x10104EA68 (0x10104EA68) (MsgRecord), reason 'RPC wait' for NSD I/O completion
```

Waiter Triage: Finding the real issue

- Waiter priority
  - Recovery: "GroupProtocolDriverThread“, "for ccMsgGroupLeave“, "for sgmMsgExeTMPhase“, "MMFS group recovery phase”
  - Local-IO: "for I/O completion on disk“, "Mount handler: for open disk device”
  - Client-IO: "waiting for exclusive use of connection“, "for getData on“,"for NSD I/O completion”
  - Revoke: "for tmMsgRevoke"
  - Others: "stealEngine loop wrapping“, "in kernel waiting to quiesces“, "waiting until pit work is complete”
  - Secondary: "change_lock_shark waiting to set acquirePending flag“, "waiting for [A-Z][A-Z] lock“, "waiting because of local byte range lock conflict“, "waiting for fetch-n-lock to complete“, "for tmMsgTellAcquire“, "wait for SubToken to become stable"

Other ways to get waiters

- mmfsadm dump waiters
- mmdiag --waiters
- mmlsnode –L –N waiters
- Use mmdsh to collect across nodes of interest

Networking

- Check networks
  - All nodes communicate over the private network
  - Security correct
  - Network PD (lost packets, netmasks same, mtu same)
    - For example the logs may show the cluster expelling nodes after correctly connecting (they don't have the correct netmask)
  - Routing correct
  - No headers! (rsh/ssh)

Management tasks

Management tasks

- Some management commands
  - mmadnode
    - Provide same details as when creating the cluster (node name, functions)
  - mmdelnode
    - Remove a node from the cluster
  - mmcrnsd
    - Create a new NSD
  - mmadddisk
    - Add a NSD to a file system
  - mmrepldisk
    - Replace a disk in a file system (cannot be used to replace a stopped disk)
  - Stop a node automatically mounting a file system
    - Stop GPFS file system mounting automatically on a node
    - touch /var/mmfs/etc/ignoreStartupMount.<file system>

Loss of Quorum

- How to recover from loss of majority for quorum nodes
- Scenario: You have 3 quorum nodes and 2 quorum nodes die
  - Nodes go into Arbitrating state
  - Add 2 new quorum nodes
    - mmchnode -–quorum –N node4,node5
    - Now you have 5 quorum nodes and 3 are running
    - Running nodes return to Active state
  - Delete missing quorum nodes
    - mmdelnode –N badnode1,badnode2
    - or
    - mmchnode –-nonquorum –N badnode1,badnode2
    - Now you have 3 quorum nodes and 3 are running
    - Running nodes remain in Active state
  - You may have to clean up nodes once they recover.

Recover from loss of primary configuration server (old config)

- How to recover from loss of a primary configuration server
- Scenario: Primary cluster configuration server fails
  - Secondary steps in
  - Reassign primary
    - mmchcluster –p newprimarynode
  - To sync up the new primary
    - mmchcluster –p LATEST
  - Add a new secondary server if required
  - Use the same process for failure of secondary cluster configuration server

Recover from loss of both configuration servers (old format)

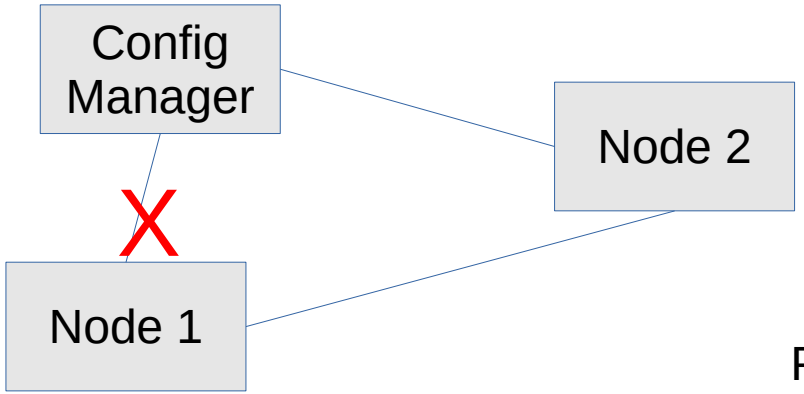
- How to recover from permanent loss of the primary configuration and secondary servers
- Scenario: Primary and secondary cluster configuration servers fail
  - Cluster Status
    - Data is online
    - mmgetstate -a no longer works
  - Reassign both node configuration servers at the same time
    - mmchcluster -p nodenewprimary –s nodenewsecondary
  - Old servers still think they are in charge until
    - mmchcluster –p Latest

Change underlying disks

- On primary Configuration Server, make a copy of /var/mmfs/gen/mmsdrfs
- Edit this file to replace old disk type with new (eg powerdisk with mpio)
  - <snip>%%home%%:60_SG_DISKS:gpfs02:4:gpfs41nsd:0:4001:dataAndMetadata:0A02E09D4CEB1155:nsd:node_1,node_2,node_3,node_4,node_5::other::powerdisk:cmd:::::::<endsnip>
- Change to:
  - <snip>%%home%%:60_SG_DISKS:gpfs02:4:gpfs41nsd:0:4001:dataAndMetadata:0A02E09D4CEB1155:nsd:node_1,node_2,node_3,node_4,node_5::other::dm-mpio:cmd:::::::<endsnip>
- Copy new mmsdrfs file to the secondary server (after taking backup)
- Replicate this fixed mmsdrfs file to the other nodes in the cluster with:
  - mmchconfig noop=yes
- Finally, stop and start the entire gpfs cluster.

Node expulsion

- How Nodes Might Get Expelled
  - Loss of Disk Lease
  - Network Problems Between Peer Nodes
- The Node Expel Sequence



Problem in network between Node 1 and Configuration manager causes Node 1 to fail to renew disk lease

Node expulsion (cont)

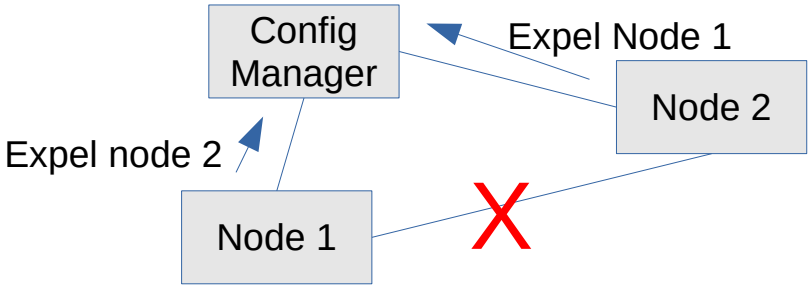
- General principles governing lease loss:
  - mmchconfig leaseDuration=xx
  - LeaseDuration:
    - Specifies how long a node is allowed to start disk I/Os after it has last been granted a disk lease (in seconds; non-positive values are invalid).
  - default: 35 seconds
  - minumum: 5 seconds
  - maximum: 30 minutes
- General principles governing lease loss:
  - mmchconfig leaseRecoveryWait=yy
  - leaseRecoveryWait:
    - Specifies how long we wait to start log recovery after a failed node's lease has run out. This means the total time to recover from a node failure is approximately leaseDuration + leaseRecoveryWait.
    - To protect file system consistency, this must be a conservative estimate of how long an I/O initiated by a failed node might take to be completed by the disk or disk controller.
  - default: 35 seconds
  - minimum: 1 second
  - maximum: 30 minutes

Node expulsion (cont)

- General principles governing lease loss:
  - mmchconfig leaseDMSTimeout=n
  - leaseDMSTimeout:
    - Specifies how long the deadman switch (DMS) will allow for pending I/Os to complete after a node's lease has expired. The purpose of the DMS is to ensure I/Os started just before a node lost its lease will complete within the time allowed by leaseRecoveryWait. If there are still I/Os pending n seconds after the lease has run out, the DMS will kill the node to prevent the device driver or host adapter from re-submitting I/Os that have not yet completed. To be effective, leaseDMSTimeout must be less than leaseRecoveryWait.
    - Example: if leaseRecoveryWait is 30 and leaseDMSTimeout is 20, all I/O must complete within 30 seconds after a node's lease runs out, but 20 seconds into the 30 second wait, the DMS on the node that lost its lease will kill the machine if it has I/Os still pending. Thus, I/Os that already made it out onto the SAN have an additional 10 seconds to reach the disk.
  - default: -1: means calculate from leaseRecoveryWait (leaseDMSTimeout = 2/3 of leaseRecoveryWait)
  - minimum: 0 seconds
  - maximum: 30 minutes

Node expulsion

- Network problems between Node 1 and Node 2 motivate them to request their counterpart be expelled.



- Nodes N1 & N2 send an expel request to the config manger when said node finds it lost contact with another node N2.
- The config manager decides which node(s) to expel.
- Chooses between the node that sent the expel request (N1) or the node requested to be expelled (N2).
  - Cases where N2 is observed having difficulty communicating with the config manager:
    - If N2 left the cluster and re-joined since N1 issued the expel request, do nothing. Assume temporary problem resolved. (Reason code 1)
    - Ignore expel request if N2 left the cluster or is marked as failed (Reason code 2)
    - If N2's disk lease is overdue, expel it. (Reason code 3)
      - Strong evidence that N2 cannot communicate with config manager.

Node expulsion

- Otherwise, the implication is: both nodes are still up. Give preference to (in order):
  - Quorum node over non-quorum node (Reason code 4)
    - Keep quorum as strong as possible.
  - Local node over remote node (Reason code 5)
    - Remote communications more likely to be less solid.
  - Manager-capable node over non-manager capable node *(Reason code 7)
    - Don't kill a token mgr or file system mgr unless we must.
  - Node managing more file systems over node managing fewer file systems * (Reason code 8)
  - All things being equal, expel the node which joined the cluster more recently. (Reason code 6)
    - Has less history of solid communications with config manager.

Node expulsion

- mmchconfig unmountOnDiskFail
  - Set to “no”
    - On disk failure, daemon marks disk as down.
    - All other nodes using the disk are informed.
    - Work proceeds as much as possible without the disk.
    - Recommended when metadata + data replication are enabled.
  - Set to “yes”
    - Disk failure will cause the local node to unmount the file system.
    - Recommended when replication not enabled.
    - Will failover to NSD servers, unless mmchdisk was used to mark the disk down.

Mysterious load?

- On some Linux distributions, cron runs the
  /etc/cron.daily/slocate.cron job
  every night.
- This will try to index all the files in GPFS. This will put a very large load on the GPFS token manager.
- You can exclude all GPFS file systems, but if indexing GPFS file systems is desired:
  - Only one node should run the updatedb command
  - Build the database in a GPFS file system, so it will be visible on all nodes when built.

Upgrades

- Sometimes an upgrade will request that you mmexportfs then mmimportfs to preserve file systems. (If customer is not carefully watching messages during this, can miss error messages and continue, wrecking the cluster).
  - Hint: always copy /var/mmfs/gen/mmsdrfs file from primary config server before doing export/import.
  - If trouble occurs, this saved mmsdrfs file can be used as input to mmimportfs, safely restoring data.
- Don't forget to change the config to reflect upgrade once all systems done
  - mmchcluster -p LATEST

Lost disk

- mmlsdisk / mmlsnsd showing device as not active
  - Perform PD and still not able to use.....
  - mmchdisk device resume -d nsdx
  - mmchdisk device start -d nsdx
- Someone claims a disk disappeared without mmdelnsd being run:

    dd if=/dev/sde2 of=nsd_head bs=512 count=1024
    strings -a nsd_head

  - Valid NSD will show

    ```
    NSD descriptor for /dev/sde2 created by GPFS Thu Jan 19 01:09:49 2009
    ```

  - Deleted NSD will show

    ```
    NSD descriptor for /dev/sde2 cleared by GPFS Thu Jan 19 01:12:42 2009
    ```

- Or use mmdiag

  ```
  mmfsadm test readdescraw /dev/hdisk4
  ```

Other issues

- NSD header re-formatted by another system
  - Use dd / mmdiag from previous slide to confirm
- Cloned nodes
  - Remove /var/mmfs/gen/mmsdrfs and add properly
- Primary and secondary config servers loose mmsdrfs
  - Issue the command:

    mmchcluster -p LATEST
  - Once all nodes have been restored, from one of the nodes issue:

    mmrefresh -f -a

Introduction to FPO

- Overview
  - IBM Spectrum Scale-FPO stands for Spectrum Scale File Placement Optimiser. It is a feature that you can enable in Spectrum Scale at the time of the creation of a file system, and allows the file system to store and use data locality properties.
  - Big data workloads that are based on the Hadoop framework must use data locality to process efficiently data by not having to transfer large chunks of data around the network. Therefore, a file system that works with data locality is a must for this kind of workload.
  - The Hadoop file system was designed for this task, but it lacks some features that are addressed by Spectrum Scale-FPO:
    - POSIX compliance
    - A redundant file system architecture for metadata processing
    - A general use file system
    - Spectrum Scale-FPO provides these features and differentiates itself from the HDFS, therefore making it a good choice for running big data workloads.

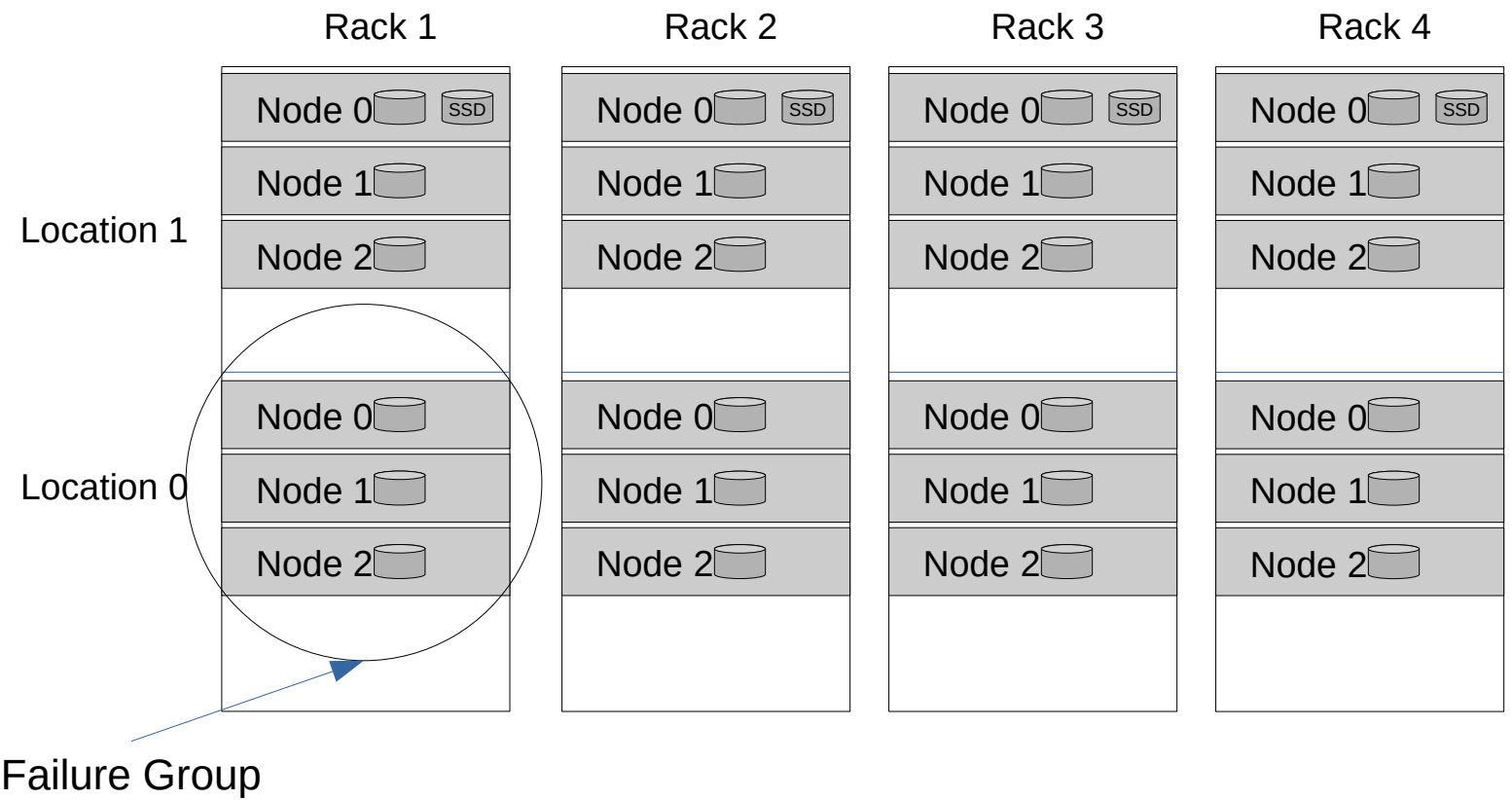
Introduction to FPO (cont)

- FPO data locality
  - Data Locality
    - In data-intensive computing, cross-switch network traffic will be the major bottlenecks for computing performance. In some computing framework, such as Hadoop Map/Reduce, the task will be scheduled to the node in which the data processed by the task resides; this technique is so-called data locality.
    - In Spectrum Scale, FPO is the only feature that supports data locality. And several options of FPO will impact the data locality, such as Write Affinity, Write Affinity Depth(WAD), WAD Failure Group(WADFG). These features will impact how the data are distributed in the whole Spectrum Scale FPO cluster.
    - Checking data locality
    - Since Spectrum Scale 4.1.0.5, a program is shipped for retrieving data locality information from Spectrum Scale FPO:
    - /usr/lpp/mmfs/samples/fpo/tsGetDataBlk.C
  - Known limits
    - Spectrum Scale FPO has the same limit as general for NSD number per file system which is maximal 2048 NSD per file system.

Introduction to FPO (cont)

- The following diagram illustrates a GPFS File Placement Optimiser (FPO) topology with four racks that each contain six nodes. Three nodes are located in the top half and bottom half of each rack. Each rack corresponds to the rack that you specify for each node when you create your InfoSphere® BigInsights® cluster. All nodes use the same hardware besides the type of disk drive that is used for data storage. In a typical GPFS FPO cluster, solid state drives (SSD) are recommended for metadata, though you can use hard disk drives (HDD) throughout your cluster.
- One node on each rack has disks that are dedicated for metadata. To have three replicas for metadata, a minimum of four failure groups are necessary to maintain the replication level in the event of single-node failure. Maintaining the maximum replication factor (three replicas) for metadata at any time is important to guarantee the maximum availability of the cluster.
- For clusters that can tolerate less replicated data in the event of a single-node failure, three failure groups might be sufficient. For larger clusters, you typically configure more than three failure groups for data.

Rack diagram (4 racks; 6 nodes / rack)



- The following table shows the failure groups for each of the racks in the previous diagram. A failure group refers to one half of the rack, and is denoted by a single-number notation for the metadata disks that belong to the system storage pool. In the previous diagram, the cluster is shown with metadata nodes in four different failure groups. The metadata failure groups correspond to the rack that they are located in.
- A three-number notation, known as the failure group topology vector, is used to denote disks that are part of a data storage pool that is configured for the GPFS FPO. The first number is the rack number (1, 2, 3, 4), the second number denotes whether the rack is in the bottom (0) or the top (1) half of the rack, and the third number denotes the position of the node in the specified half of the rack. For example, in relation to the previous diagram, the notation 2,1,0 refers to the second rack, top half, node 1. All nodes using disks in a given half of a rack belong to the same failure group. For example, the three nodes with disks using the failure group topology vectors (2,1,0), (2,1,1), and (2,1,2) are in the (2,1) failure group. In the previous diagram, the failure groups for the data disks are: (1,0), (1,1), (2,0), (2,1), (3,0), (3,1), (4,0), (4,1).
- The configuration in the diagram illustrates two failure groups per rack, which combine for eight total failure groups for this cluster. Each failure group contains three nodes.

Failure groups (fpo)

Rack	Failure group (lower)	Failure group (upper)
1	1,0,0 1,0,1 1,0,2	1,1,0 1,1,1 1,1,2
2	2,0,0 2,0,1 2,0,2	2,1,0 2,1,1 2,1,2
3	3,0,0 3,0,1 3,0,2	3,1,0 3,1,1 3,1,2
4	4,0,0 4,0,1 4,0,2	4,1,0 4,1,1 4,1,2

ESS

- Elastic Storage Server (ESS) packaged for customers
- ESS behind the marketing

High performance server based storage

- Building block approach, scale capacity (starting from 50TB grow to 100's of petabytes) and performance while keeping the single namespace
- Benefits
  - High Performance and Density
    - Today’s workloads demand Fast Access to Petabytes of Data
    - Accelerates current data workloads while creating future-proof infrastructure
    - Complete Petascale storage in a single rack, including servers, disks, and middleware
    - Optimised for multi-workload access including Cloud, Analytics, Media, and HPC
  - Flexibility
    - Scalable Growth – start small and grow easily in a building block approach
    - Encryption available for highly secure data and multi-tenant access
    - Optimise around performance and capacity with SSD, SAS, and NL-SAS drives
    - Scalable & extendable as needs change and the enterprise grows

High performance server based storage (cont)

- Benefits (cont)
  - Data Protection and Availability
    - Declustered RAID technology to reduce disk rebuild times up to 7 times
    - Complete Path Data Integrity Protection all the way from Disk Surface to Client
    - Hierarchical Storage Management to move unused data to lower cost storage devices
    - Active File Management for off-site data replication for local access and disaster recovery
  - Building blocks
    - Platform and storage management console (S812L) and IBM Elastic Storage Server node (S822L)
    - Storage for 5146-GLx models (DCS3700 Expansion Unit) for 5146-GSx models (5887 disk drive enclosure
    - IBM Rackswitches (G7028, G8052, G8264); HMC; Console and Rack

What niche is it aimed to fit?

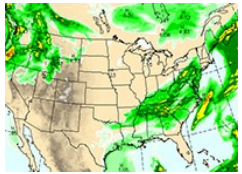
- Scalability to the hundreds of Petabytes and hundreds of GB/Sec
- A Single Name Space Across Entire File System
- A Building Block approach to Scale as Needs Grow
  - Start Small – Grow over time
  - Scalable Performance as Storage Blocks are added
- An Actual Instance of Software Designed Storage
- Based upon the Industry Leading Power Architecture, Elastic Storage Software, and off-the-shelf JBOD enclosures
- A Scalable Common Foundation for Analytics, File Serving, Object Storage
  - A common Data Lake for workload variety without the need for Data Islands

What niche is it aimed to fit (cont)?

- New generation applications have some important differences to traditional applications
  - Traditional applications
    - Transactions and processes
    - Controlled data growth
    - Efficiency through virtualisation
  - New applications
    - Insights and engagement
    - Rapid pace and massive data scale
    - Global Elasticity

Examples

- Low latency global access to data
- Linear scale out capacity and performance
- Enterprise storage services on standard hardware



NCAR

Climate and weather modeling with 11 Petabytes on line and 14 Petabytes archive on tape

4 time Champion Infiniti Red Bull Racing does real-time race analytics

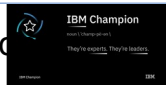
Vestas

Wind turbine design analysis done in hours instead of weeks

SPEICHER M1
MEDIA ASSET MANAGEMENT

Private Cloud for digital media enables global collaboration for film productio

IBM Belisama



Examples

- Serving the web
  - Perform I/O directly to storage
  - High space efficiency
  - Failover
  - Low Cost Archive w/ Tape Integration
  - Requires no changes to Swift
  - Native support for enterprise features
- Serving MapReduce applications
  - Take advantage of performance of GNR, and end to end integrity checks
- Serving hyper data needs (gpfs throughput 400GB/s at Argonne national labs)
  - Argonne Labs achieved this throughput using GNR