LCI workshop

on high performance clustered computing

Linux Clusters Institute: High Performance Storage

University of Wyoming, 22-26 May 2017

Pamela Hill, National Center for Atmospheric Research

pjghill@gmail.com

NCAR NATIONAL CENTER FOR ATMOSPHERIC RESEARCH

UW

HPC Storage, Part 1

HPC Storage Concepts, Planning and Implementation

Targets for Session #1

Target audience:

Those involved in designing, implementing, or managing HPC storage systems

Outline:

- Important Concepts & Terminology
- Goals & Requirements
- Storage Hardware
- File Systems
- Wrap Up

Important Concepts & Terminology

What is Storage?

A place to **store** data

- Cache Memory
  - Fastest access, closest to CPU
- Dynamic Random Access Memory (DRAM)
  - Close to CPU, generally used during computation
- Solid State Disk
  - Can be internal to the system or part of an external storage system
- Spinning Disk
  - Can be internal to the system or part of an external storage system
- Tape
  - Typically located in a tape library

HPC Storage has generally been considered to be high-performance spinning disk

- This is changing somewhat due to innovate memory and solid state storage solutions

Important Concepts & Terminology

- ***IOPs:*** Input/Output Operations per second
- ***RAID:*** Redundant Array of Inexpensive Disk
- ***JBOD:*** Just a Bunch of Disk
- ***RAS:*** reliability, accessibility, serviceability
- ***Storage Server:*** provide direct access to storage device and functions as a data manager for that disk
- ***Storage Client:*** accesses data, but plays no role in data management
- ***LAN:*** local area network
- ***WAN:*** wide area network
- ***SAN:*** storage area network

Important Concepts & Terminology

- ***High Availability (HA)***
  - Components are configured in failover pairs
  - Prevents a single point of failure in the system
  - Prevents a service outage
- ***Failover Pairs***
  - Active/Active
    - Both component share the load
    - On failure one component takes over the complete load
  - Active/Passive
    - One component services requests, the other is in standby
    - On failure the standby becomes active
- ***Networks***
  - InfiniBand (IB)
  - OmniPath (OPA)
  - Ethernet (TCP/IP)
- ***Host Connectivity***
  - Host Bus Adapter (HBA)
  - Network Interface Card (NIC)

Important Concepts & Terminology

- ***Raw Space:*** what the disk label shows. Typically given in base 10.
  - 10TB (terabyte) == 10*10^12 bytes
- ***Useable Space:*** what "df" shows once the storage is mounted. Typically given in base 2.
  - 10TiB (tebibyte) == 10*2^40 bytes

Useable space is often about 30% smaller than raw space

- Some space is used for RAID overhead, file system overhead, etc.
- Learning how to calculate this is a challenge
- Dependent on levels of redundancy and the file system you choose

Goals & Requirements

Which Storage Architecture Is Best?

- The short answer: The one the solves all your problems
- The long answer: There is no single answer to this question
  - Each is designed to serve a specific need
  - Each works very well when built and deployed according to their strengths
  - Application characteristics define which is the best choice
  - Issues to consider
    - Application requirements
    - User expectations (in their own words)
    - Budget constraints
    - Expertise in support team
    - Existing infrastructures
    - How the product implements a specific architecture
- Compromise between competing needs is almost always the end result

Storage System Design Goal

In an ideal world, the I/O subsystem will be perfectly balanced

- All components contribute equally to the overall performance of the system
  - Overburdening one component in the system (e.g., HBAs) will degrade overall system performance.

This is not an ideal world

- Common imbalances
  - Capacity is more important than bandwidth; the number of disks may exceed the aggregate bandwidth supported by the controllers, back-end channels or HBAs
  - Overall aggregate bandwidth of the storage system exceeds the network capacity of the computational system
  - Budget is typically a driver towards imbalance

“Performance is inversely proportional to capacity.” – Todd Virnoche

Requirements Evaluation

- Stakeholders
  - Computational Users
  - Management
  - Policy Managers
  - Funding Agencies
  - System Administration Staff
  - Infrastructure Support Staff
  - IT Security Staff
- Usage Patterns
  - Write dominate
  - Read dominate
  - Streaming I/O vs Random I/O
- I/O Profiles
  - Serial I/O
  - Parallel I/O
  - MapReduce I/O
  - Large Files
  - Small Files
- User Profiles
  - Expert vs. Beginner
  - Custom vs. commercial application
- Infrastructure Profile
  - Integrated with HPC resource
  - Standalone storage solution
  - Network connectivity
  - Security requirements

Gathering Stakeholder Requirements

- Who are your stakeholders?
  - What features are they looking for?
  - How will people want to use the storage?
  - What usage policies need to be supported?
- From what science/usage domains are the users?
  - What applications will they be using?
- How much space do they anticipate needing?
- Can they define the performance characteristics they need?
- Are there expectations of access from multiple systems?
- What is the distribution of files?
  - Sizes, count
- What is the typical I/O pattern?
  - How many bytes are written for every byte read?
  - How many bytes are read for each file opened?
  - How many bytes are written for each file opened?
- Are there any system-based restrictions?
  - POSIX conformance - do you need POSIX interface to the file system
  - Limitations on number of files or files per directory
  - Network compatibility (IB, OPA, Eth)

Application I/O Access Patterns

- An application's I/O transaction size and the order they are accessed in defines an application's I/O access pattern. This is a combination of how the application does I/O along with how the file system handles I/O requests.
- For typical HPC file systems, sequential I/O of large blocks provides the best performance. Unfortunately, these types of I/O patterns aren't the most common.
- Understanding the I/O access patterns of your major applications can help you design a solution your users will be happy with.

Common Application Access Patterns

- Streaming (bandwidth centric)
  - Records are accessed only once, file is read/written from beginning to end
  - File size are generally quite large (e.g., GB or more)
  - Performance is measured in overall bandwidth (e.g., MB/s, GB/s)
  - File operations are minimal
  - Most common in digital media, HPC, scientific applications
- I/O Processing (IOP centric)
  - Small transactions (i.e., less than FS block size)
    - Small files, randomly accessed
  - File operation counts can be high
  - Common examples: bio-informatics, rendering, home directories
- Transaction Processing (IOP centric)
  - Small transactions (i.e., files or record less than the block size), but often displaying good temporal locality
    - Databases tend to be efficient due to data being in contiguous blocks
  - File operation counts can be high
  - Common examples: commercial applications, databases

HPC I/O Access Patterns

- Traditional HPC
  - Streaming large block writes (low IOPs rates)
  - Large output files
  - Minimal metadata operations
- More common today
  - Random I/O patterns (high IOPs rates)
  - Smaller output files
  - Large number of metadata operations

Challenges

- Choosing a block size that fits your application I/O pattern
- GPFS breaks blocks into 32 sub-blocks, so you can match your sub-block size to your average file size
- IOPs becomes more important with Random I/O patterns and small files

Gathering Data Requirements

- Do you need different tiers or types of storage?
  - Active long-term (project space)
  - Temporary (scratch space)
  - Archive (disk or tape)
  - Backups (snapshots, disk, tape)
  - Encryption
- Data Restrictions
  - HIPAA and PHI
  - ITAR
  - FISMA
  - PCI DSS
  - And many more (SOX, GLBA, CJIS, FERPA, SOC, …)
- Ingest/Outgest
  - Data transfer characteristics

Training & Support Requirements

- Training
  - Sys Admin Staff
    - How much training does your staff need?
    - Vendor supplied training?
    - Does someone on your staff have the expertise to provide training?
  - User Services
    - How much training does your users support staff need?
    - Does you user support staff have the expertise to provide user training?
  - Users
    - How much training will your users need to effectively use the system?
    - How often will training need to be provided?
- System Support
  - Does the vendor provide support for all components of your system?
  - Do support for parts of your system come from the open source community?
  - What are the support requirements for your staff?
    - 7x24
    - 8x5 M-F
  - Do you have Service Level Agreements (SLAs) with your user community?

Common Storage Usage

- Temporary storage for intermediate job results
  - Typical ‘scratch’ usage
- Active long-term storage for runtime use
- Backups
- Archival
- Virtual machine hosting
- Database hosting
- Data ingestion

- Data pre-processing
- Data post-processing
- Data serving
  - Data portals
  - Web services
  - Data transfer services (DTNs)
  - NFS/CIFS
  - Centralized software repositories
- System Administration Storage
  - Log files
  - Monitoring
  - Cluster management tools

Common Design Tradeoffs

- Aggregate Speed or Bandwidth
- Capacity
- Scalability
- Cost/Budget
- Physical Space
- Environmental Needs
  - Power, cooling, etc.
- Reliability/Redundancy Features
- Sys Admin features
  - Management tools
  - Monitoring
  - Vendor support
  - Community support

Storage Hardware

Storage Characteristics

- Controllers
- Host Connections
- Chassis
- Drawers/Trays
- Disk Channels
  - SAS, SATA Protocols
- Disks
  - Spinning
  - Solid state
  - SAS, SATA, NVMe Protocols
  - JBOD
  - RAID
- Block Storage
- Object Storage
- Networks
  - SAN, LAN, WAN
- Tape Drives
- Tapes
- Disk Cache

Storage Evaluation

- System Features
  - Data integrity features
    - RAID, erasure encoding, bit correction, parity check on read
  - Data security features
    - Encryption, Compartmentalization
  - Expandability
    - Capacity & bandwidth
  - Floor space/environmental features
  - Disk types supported
    - Mixed types supported in same system
  - Networks supported
    - Storage to file system server
    - File system server to client
- Performance Features
  - Aggregate bandwidth
  - Streaming bandwidth
  - IOP rate
  - Failed disk rebuild times
- User Interfaces
  - POSIX based file system
  - Object based file system (REST API)
- Redundancy Options
  - Server failover (HA)
  - Redundant data path to disks
  - Dual controllers w/ failover
    - active/active or active/passive
- System Tools
  - Deployment tools
    - Low level formatting
    - LUN configuration
  - Maintenance tools
    - Take component offline for repair
    - Phone home feature
    - Health monitoring
  - Management Tools
    - Performance monitoring
    - Usage monitoring

Networks

- Distributed/Parallel File Systems require large stable networks
- InfiniBand (IB), OmniPath (OPA), Ethernet, Proprietary
- Redundancy/Failover
- Routing
  - LNET, TCP
- Connectivity into Computational Clusters
- Remote Direct Memory Access (RDMA)
- RoCE (RDMA over Converged Ethernet)

Comparing LAN and SAN Topologies

LAN Topology

- User data, metadata, file system overhead tasks all traverses LAN fabric
- Disks attach only to servers
- Applications generally run only on the clients
- Economically scales out to large clusters
- ***Potential bottleneck:*** Network Interface Adapters (NICs)

SAN Topology

- User data and metadata traverse SAN fabric
- Overhead data traverses the LAN fabric
- Disks attach to all nodes in the cluster
- Applications may run on all nodes in the cluster including storage server nodes
- Works well for small clusters
- ***Potential bottleneck:*** Host Bus Adapters (HBAs)
  - Cost of fabric makes this too expensive to scale out to large clusters
  - Ideal for small clusters such as large shared memory systems

Block Storage vs. Object Storage

Block Storage

- A block holds a chunk of data, files typically reside in multiple blocks.
- Data is accessed through a request to the block address
- The file system controls access to blocks, imposes a hierarchical structure and defines and updates metadata
- File systems are generally POSIX compliant
- Access is through the OS layer, user has direct access to the file system
- Storage systems are generally feature rich and externally based
- Raw block storage is common in large scale database applications allowing direct block access

Object Storage

- An object hold both data and it's associated metadata
- Data is access through a request for the object ID
- Objects are stored in a flat structure
- Metadata can be flexible and defined by the user
- Resiliency is provided through multiple copies of the object, generally geographically distributed
- Access is through a REST API application not the OS
- Systems are generally built with commodity servers with internal disks

Strong Consistency vs. Eventual Consistency

Strong Consistency

- Block storage systems are *strongly consistent*
- Typically used for real-time processing such as transactional databases
- Good for data that is constantly changing
  - Updating a file only requires changing the blocks that have changed
- Limited scalability especially within a geographically distributed system
- Guarantees that a read request returns the most recent version of data

Eventual Consistency

- Object storage systems are *eventually consistent*
- High availability for data that is relatively static and rarely altered
  - Updating an object requires a re-write of the entire object
- Typical uses are for multimedia or unstructured data
- There is no guarantee that a read request returns the most recent version of data

Network Attached Storage (NAS)

Appliance Concept

- Integrated storage solution
  - Servers, storage controllers, disks, networks, file system, protocol, etc.
  - Main advantage: “black box” design
    - Minimal experience to administrate
    - Vendor configured, works well if it meets your requirements, hard to alter
  - Not intended for high performance storage
- Typically provides an NFS server and/or CIFS/Samba solution
  - Solid server side solution, does *not* improve client side access or performance
  - May support other protocols (e.g., iSCSI, http, S3)
- Generally based on TCP/IP Ethernet LANs
- Generally high IOPs configurations

Redundant Array of Inexpensive Disk (RAID)

RAID is a data storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy, performance improvement, or both. [1] RAID can be implemented either in hardware or software.

RAID Set: a grouping of disks which contain user data, parity information and occasionally metadata

- Different OEM vendors use different names for this grouping of disks
  - DDN: Tier
  - IBM: Array
  - NetApp: Array

Parity: information used to reconstruct user data when a disk failure occurs

LUN: A logical unit that presents the to Operating System (e.g., a /dev/ entry for Unix/Linux Oss

Best Practice: LUNs vs RAID sets

- In general you will configure 1 LUN per RAID set. Some vendors have designed their system to handle this a different manner. Always follow the vendor recommendations for the storage system/file system configuration.

[1] Wikipedia

Common RAID Levels

- Standard Levels
  - RAID 0: striping without mirroring or parity
  - RAID 1: data mirroring, without parity or striping
  - RAID 2: no longer used
  - RAID 3: byte-level striping with dedicated parity disk
  - RAID 4: block-level striping with dedicated parity disk(s)
  - RAID 5: block-level striping with distributed parity
    - No loss of data with a single disk failure
    - Data loss occurs with third failure
  - RAID 6: block-level striping with dual distributed parity
    - No loss of data with two disk failures
    - Data loss occurs with third failure
    - Most common in larger systems today
- Hybrid Levels
  - RAID 0+1: create two stripes then mirror
  - RAID 1+0: stripping across mirrored drives

Common Enterprise Class Drive Technologies

- SAS (Serial Attached SCSI)
  - 15K rpm, 10K rpm, 7.2 rpm, 12Gb/s interface
  - Fastest drives available, seek times are substantially higher
  - Highest reliability, typically has a higher MTBF rate
  - Higher signaling rate, therefore longer cables are possible
  - Great for metadata
- NL-SAS (Nearline SAS)
  - 10K rpm, 7.2K rpm, 12Gb/s interface
  - SAS interface w/ SATA head and media
  - Dual ported for redundant paths
  - Full SCSI command set
  - No SATA translation necessary
- SATA (Serial Advanced Technology Attachment)
  - 7.2K rpm, 5.4K rpm, 6Gb/s interface
  - Capacities can be much larger
  - Price is typically lower
  - Typically has a lower MTBF rate

Solid State Storage Technologies

- SSD
  - MLC NAND based flash memory is most common
  - Faster access times with lower latency, high IOPs
  - Smaller capacity than HDDs
  - Higher cost than HDDs
  - Need to be aware of wear patterns (duty cycle)
    - Lots of re-writes will wear out spots in the media
    - Versions available for read dominate applications and mixed read/write applications
- Interfaces Supported
  - SAS
  - SATA
  - PCIe
  - NVMe
    - Packaged as a PCIe card or 2.5” form-factor drive with U.2 connector
    - U.2 connector provides four-lanes of PCIe

Newer Disk Technologies

- SMR (Shingled Magnetic Recording)
  - Writes data in overlapping areas, like shingles
  - Write heads are larger than read heads allowing for higher density
  - Writes are slower since a shingle has to be “lifted” to re-write and underlying shingle
  - Good for archival storage and data that is static
- HAMR (Heat-Assisted Magnetic Recording)
  - Uses a laser for writes instead of magnetic heads
  - Targeting larger capacities and faster access times
  - No current product on the market

Common HPC Solution

- Block storage
  - Most common drive types are block devices
  - Most systems provide either hardware or software RAID
  - SSD for metadata or frequently read static data
  - HDD for main data storage
- File system hybrid
  - POSIX interface presented to the user
  - Object based structure for storing data on block devices
  - Metadata may be internal or external to the data
  - Metadata generally supports attributes beyond basic POSIX attributes

File Systems

File System Characteristics

- POSIX
- Object Store
- Serial/Parallel
- Journaling
- Disaster Recovery
- Archival
- Snapshots
- Policy Engine
- Encryption
- Compression/De-Duplication
- Global name space
- Client/Server
- Data/Metadata
- Shared Disk File System
- Distributed File System
- Clustered File System

Clustered File System Types

Clustered File System: data is shared between systems by mounting the file system on multiple clients. Parallel file systems are a type of clustered file system where there are multiple storage nodes serving the file system over a network.

Shared Disk

- Multiple systems all have access to the same external disk subsystem at the block level
- Generally utilizes a SAN network model

Distributed

- Client/Server model
- Client accesses disk blocks through a network request to a server

Parallel

- Data is distributed among multiple server nodes
- File system handles locking and consistency

Distributed File System Design Goals [1]

- ***Access Transparency:*** clients are unaware that they are access remote data
- ***Location Transparency:*** clients are unaware of the actual location of data
- ***Concurrency Transparency:*** all clients have the same view of the data state
- ***Failure Transparency:*** clients continue to operate correctly after a server failure
- ***Heterogeneity:*** clients/servers can be of different hardware and operating system
- ***Scalability:*** file system should work as well at small client node counts as at large client node counts
- ***Replication Transparency:*** any data replication should be invisible to clients
- ***Migration Transparency:*** any data migration should be invisible to clients

[1] https://en.wikipedia.org/wiki/Clustered_file_system

File System Evaluation

- Space Management
  - Policy Engine
    - Placement
    - Data Migration
    - Purge Policies
  - Accounting
  - Snapshots
  - Encryption
  - Compression/De-Duplication
  - Containers, Filesets
  - Quotas
    - User/Group
    - File System Level
    - Directory/Container Level
- File System Management
  - Dynamically Grow/Shrink File System
  - Dynamically Grow/Shrink inode space
  - Mixed software release levels
  - Ability to fence/expel problem clients
- Storage Cluster Management
  - Centralize control
  - Remote management
  - Rolling/Phased Upgrades
    - OS Updates
    - File System Updates
    - Firmware Updates
  - Configuration management
    - Consistent software
    - Consistent configuration
- System Monitoring/Notification
  - Centralize logging
  - Performance monitoring
  - File and metadata transactions
  - System software events
  - Hardware events
  - Integration with existing tools
    - Nagios, Ganglia, etc.

Serial vs. Parallel File Systems

Serial

- It doesn't scale beyond a single server
- It often isn't easy to make it reliable or redundant beyond a single server
- A single server controls everything

Parallel

- Speed increases as more components are added to it
- Built for distributed redundancy and reliability
- Multiple servers contribute to the management of the file system

Common HPC File Systems Solutions

Most Common

- NFS (Serial NAS appliances, re-exported parallel file system)
- IBM Spectrum Scale (GPFS)
- Lustre
- Panasas/PanFS

Less Common

- OrangeFS
- Gluster
- Ceph
- XtreemFS
- CIFS
- HDFS
- Swift

Network File System (NFS)

Characteristics

- Can be built from commodity parts or purchased as an appliance
- A single server typically controls everything
- Storage is directly attached to the servers
  - No redundancy or failover
- Each server controls only the data it has direct access to
- Functionality built into all major Linux distributions
- Supports multi file systems (ext3, ext4, zfs, xfs …)

System Components

- File System Server
- Server attached storage

NFS Architecture

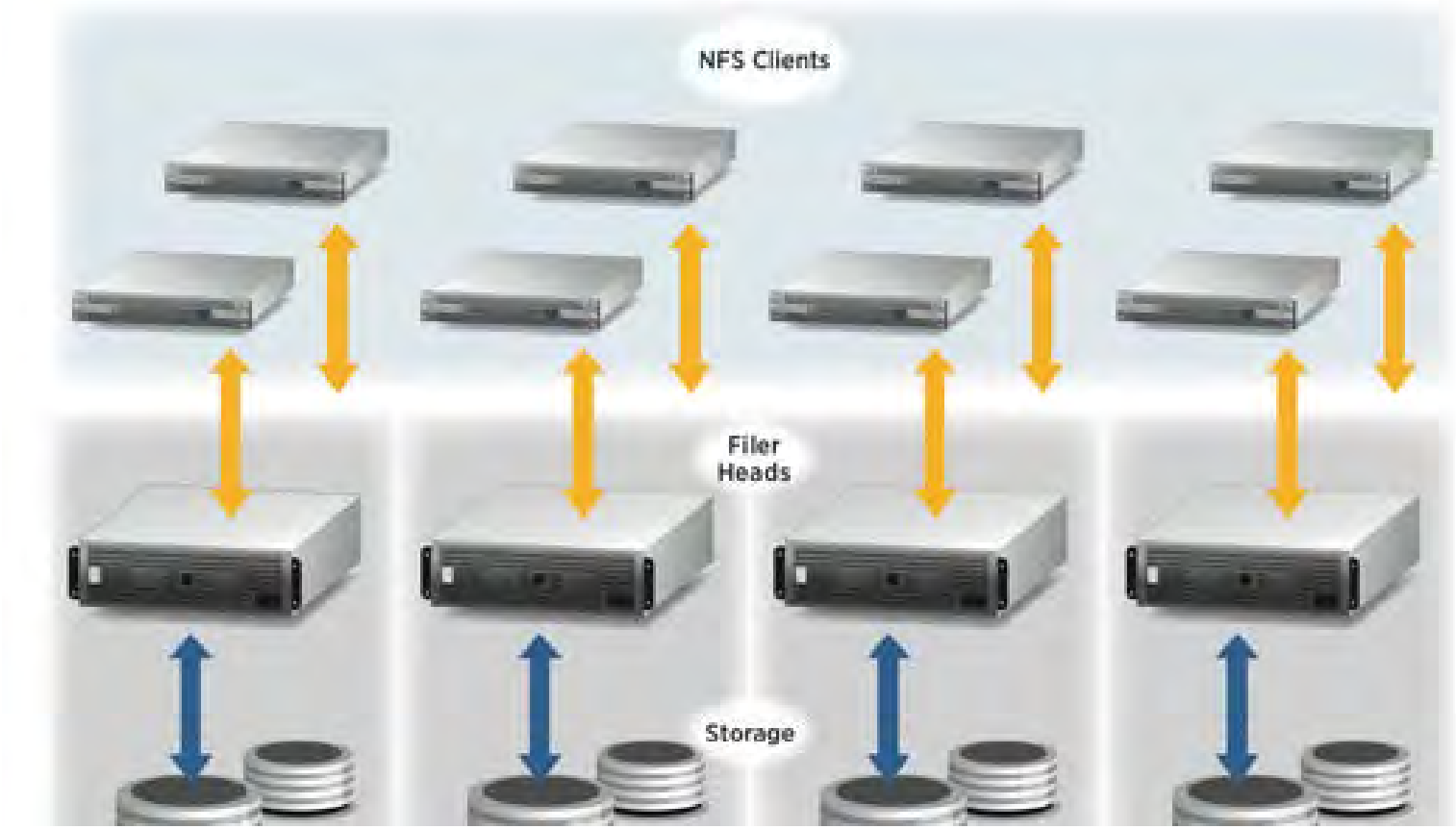


Image credit: www.panasas.com

Characteristics

- All user data is accessible from any disk to any node
- Metadata may be shared from all disks or from a dedicated set of disks
- Supports multi copies of metadata
- All data movement between nodes and disk is parallel
- Large number of nodes utilize the file system

System Components

- Server Nodes - Cluster Manager, Quorum Nodes, File System Manager
- NSD Servers – direct or SAN attached to physical storage, block access
- Network Shared Disk (NSD) – LUNs formatted for GPFS usage

Sources

- IBM – www.ibm.com/systems/storage/spectrum/scale/

IBM Spectrum Scale

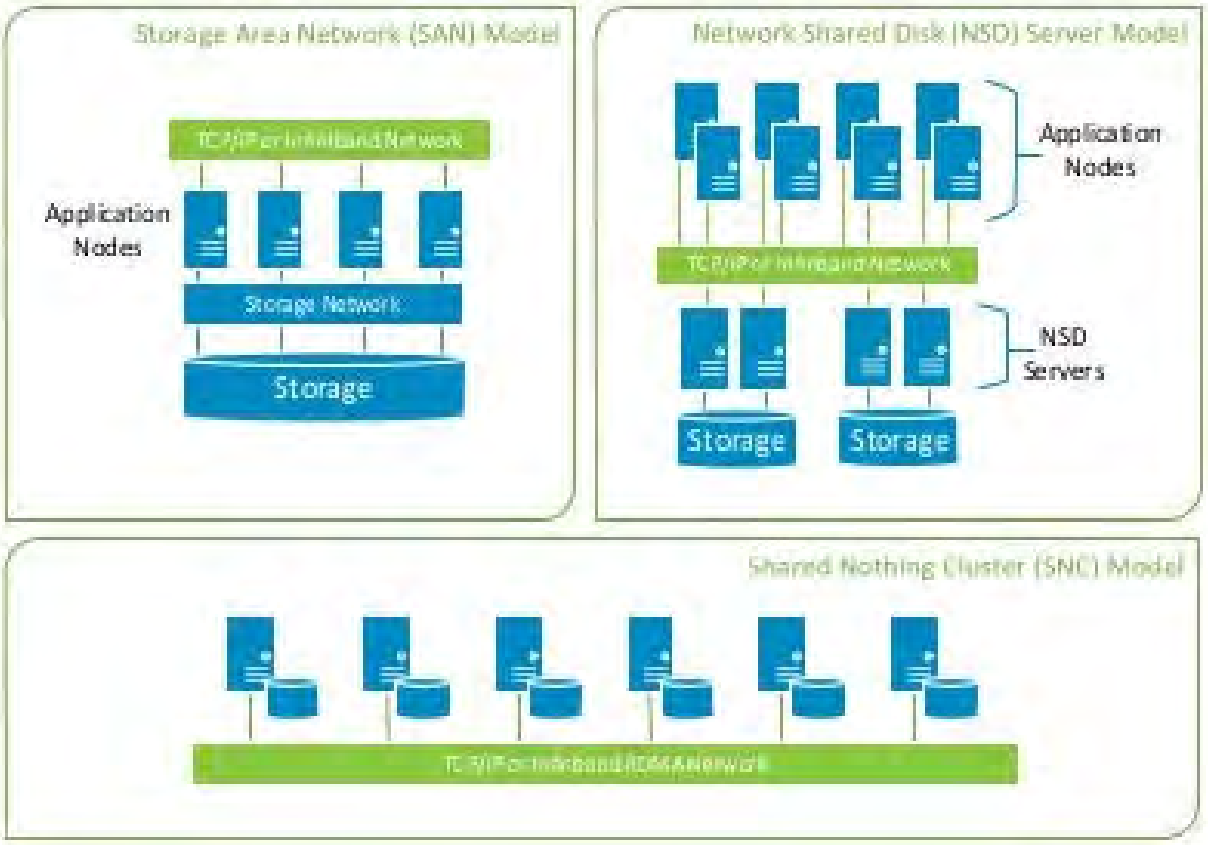


Image credit: www.ibm.com

lustre™

Characteristics

- Open source parallel distributed file system
- Metadata services and storage are segregated from data services and storage

System Components

- Metadata Servers (MDS) – metadata servers, client access is through the MDS
- Metadata Targets (MDT) – dedicated metadata storage
- Management Servers (MGS) – Lustre cluster management
- Management Targets (MGT) – management data storage
- Object Storage Servers (OSS) – data servers, client access is through the OSS
- Object Sever Targets (OST) – dedicated object storage

Sources

- OpenSFS – nonprofit organization to advance Lustre development and ensure it remains vendor-neutral and open source.
- lustre.org – community support for Lustre source
- Intel – packages an Enterprise addition of Lustre along with support

lustre™

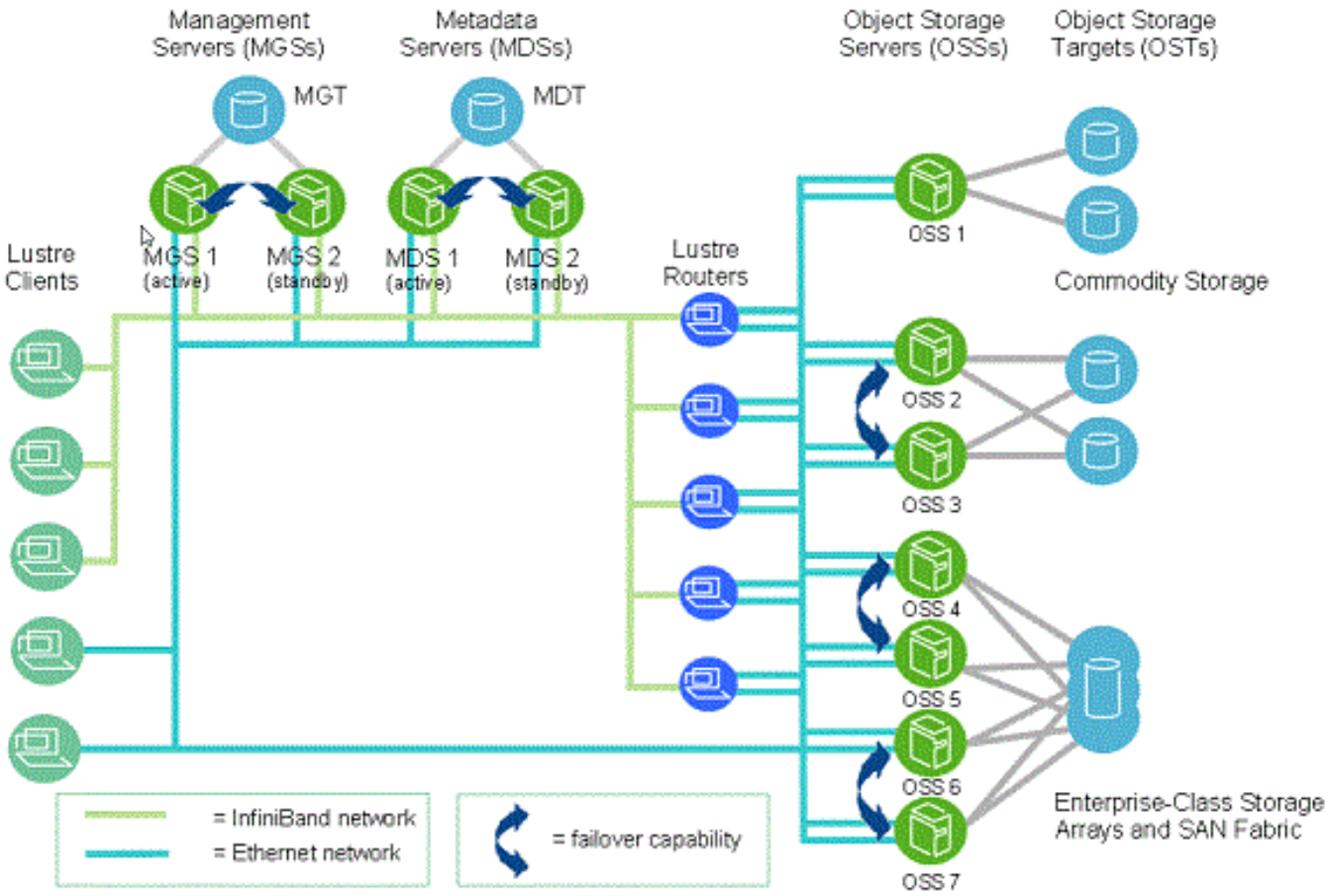


Image credit: www.opensfs.org

Characteristics

- A hybrid NAS solution supporting high parallel performance
- Linear scaling of performance as capacity grows
- Panasas File System (PanFS) relies on RAID 6+ triple parity for reliability
- Appliance style management system, scaling without increased management load
- Object based storage system
- Internally multi-tier'd for performance
  - Tier 1: Fast RAM for caching
  - Tier 2: Flash Storage for metadata and small files
  - Tier 3: Enterprise SATA drives
- Data management between tier's is handled within the Panasas software

System Components

- Shelf/Switch – holds blades and network switches (10GbE)
- Storage Blade – holds SSD and SATA drives
- Director Blade – controls data management and virtualizes data objects across systems

Sources

- www.panasas.com

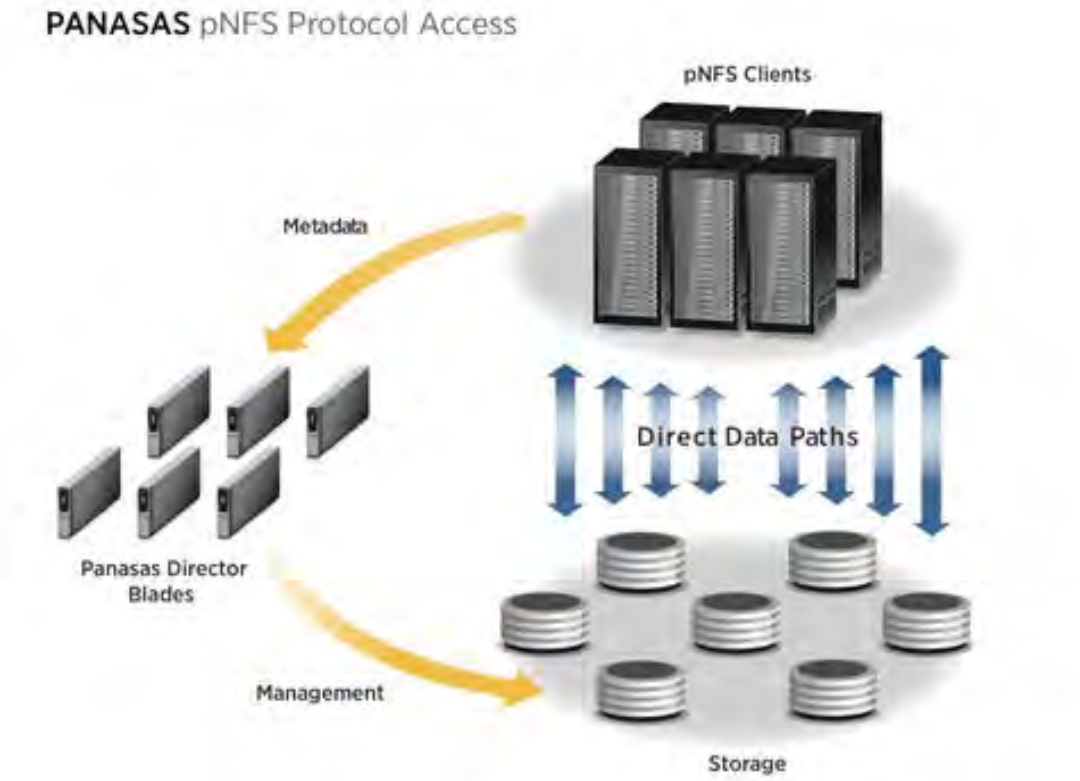


Image credit: www.panasas.com

Appliances

- Appliances generally come with vendor tools for monitoring and management
- Turn-key solution
- Vendor controls some of the configuration options
- Can be less flexible for site customization
- Can be easier for staff to manage

Screenshot of Panasas management tool

Image credit: www.panasas.com

Common HPC Appliance Solutions

- Lustre Appliances
  - DDN EXAScaler
  - Seagate ClusterStor L Series
  - Cray Sonexion
  - Dell/EMC HPC Lustre Storage
- Spectrum Scale Appliances
  - DDN GRIDScalar
  - IBM ESS
  - Lenovo GSS
  - Seagate ClusterStor G Series
- Panasas
  - PanFS
  - Proprietary

System Support

	Server	Client	NFS/CIFS	Ethernet	IB / OPA
Lustre	Linux	Linux	Yes	10-100 GbE	Yes
Spectrum Scale	AIX, Linux	AIX, Linux, Windows	Yes	10-100 GbE	Yes
Panasas	Linux	Linux, OS X	Yes	10 GbE	No
Ceph	Linux	Linux			
OrangeFS	Linux	Linux, Windows	Yes	Yes	Yes
XtreemFS	Linux	Linux, Windows, OS X		Yes	Yes
Gluster	Linux	Linux	Yes	Yes	Yes
Hadoop	Linux	Linux	No	No	No

Wrap Up

Conclusions

- There is no one right solution for HPC Storage systems
- Each solution is highly dependent on your environment
- The more time you spend gathering requirements and talking with your stakeholders, the better your design
- Trade-offs are inevitable, focus on prioritizing basic features
- Avoid ‘Gold Plated’ solutions
- Without careful design, I/O can seriously degrade parallel efficiency (e.g., Amdahl’s law)
- Good I/O performance requires hard work, careful design and the intelligent use of your chosen file system
- I/O is not the entire picture; improving I/O performance will uncover other bottle necks

HPC Storage, Part 2

Case Studies, Best Practices, Hints & Tips, Benchmarking, Non-Traditional Storage, Futures

Targets for Session #2

Target audience:

Those involved in designing, implementing, or managing HPC storage systems

Outline:

- Case Studies
- Best Practices, Hints & Tips
- Benchmarking
- Non-Traditional File Systems
- Futures

Case Studies

Data Transfer Service

Scenario

- User needs to transfer a lot of data from/to your site

Questions

- Where is the data coming from/going to?
- What is the network infrastructure between systems/sites?
- Are there any security restrictions on the networks?
  - Firewalls, Encryption required
- What type of data is it?
  - Lots of small files, few large files
- Is this a one time occurrence or a reoccurring transfer?
- What data transfer protocols exits on both ends?
- Is there a budget associated with this request?

Possible Solution

- Data Transfer Service

Data Transfer Service Architecture

- Science DMZ
  - Network configuration for allowing data transfers to occur outside the bounds of organizational firewalls
- Data Transfer Nodes
  - Configured on the Science DMZ
  - Secured to only handle data transfers
  - Has a fast path to storage where data resides
    - e.g. mounts your parallel file system
  - Multiple nodes can support parallel data transfers
  - Bandwidth is limited by network pipes
  - Supports Globus data transfer services

Example - Globus Network Paths

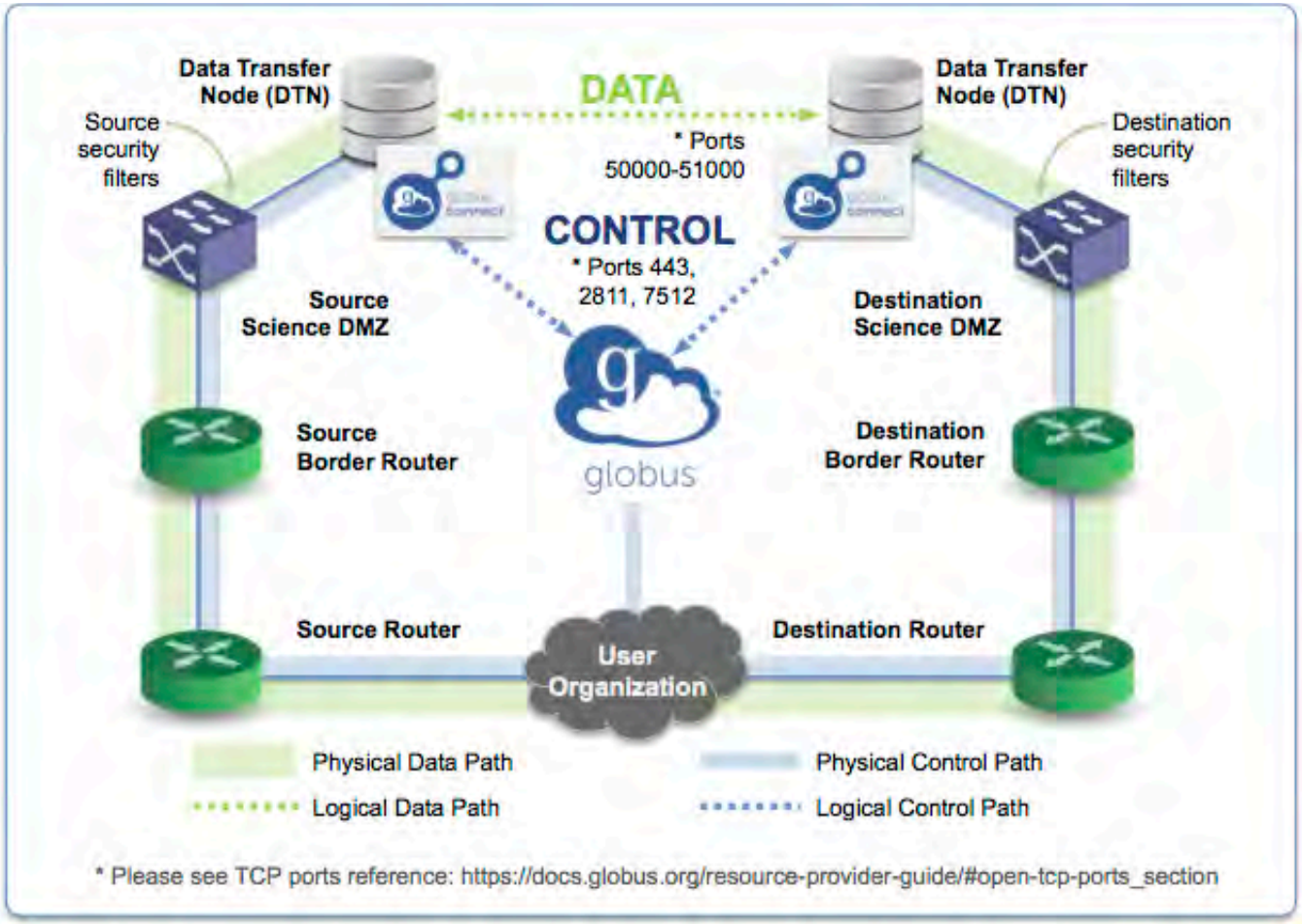


Image credit: www.globus.org

Example - Globus SaaS

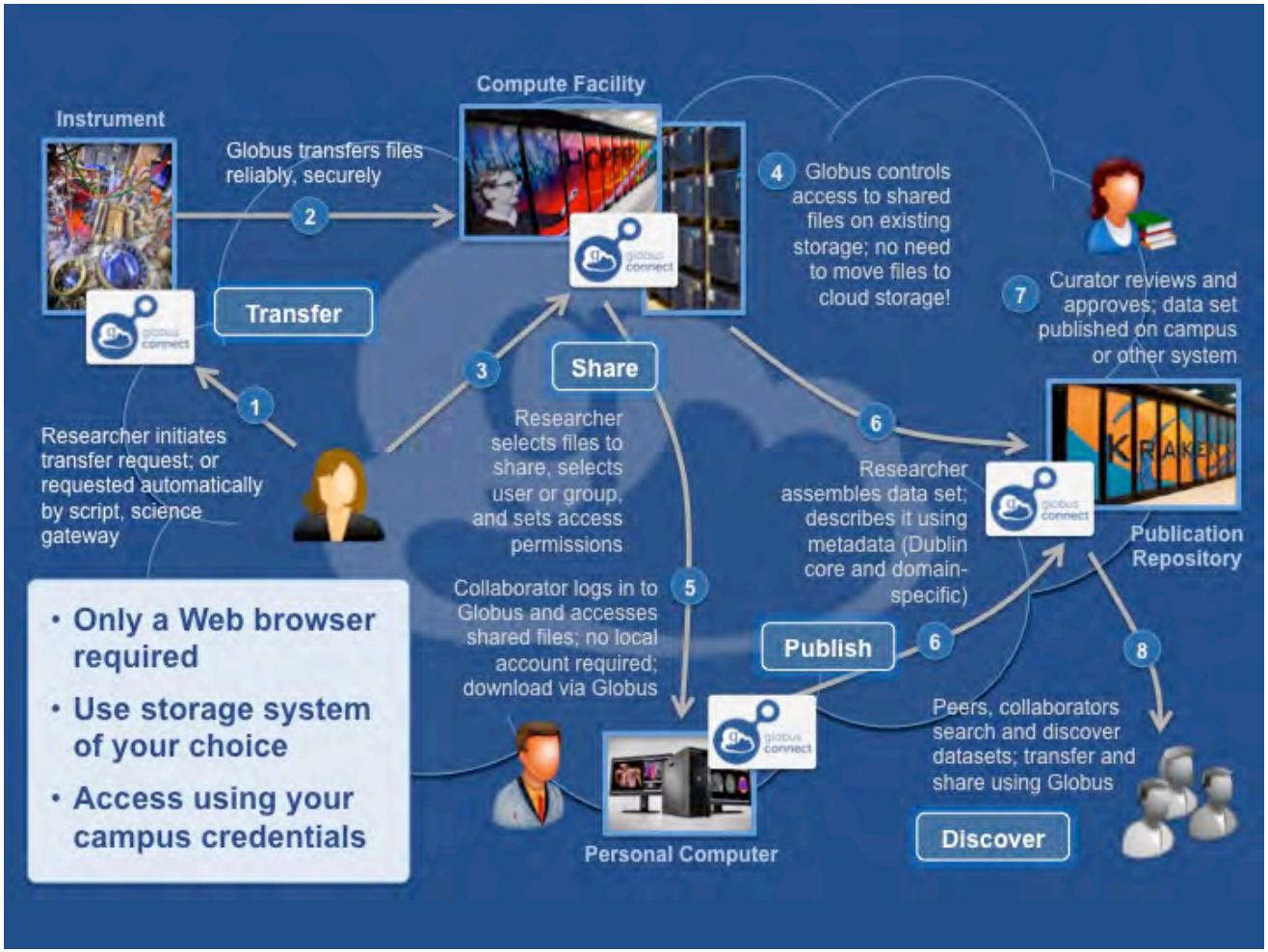


Image credit: www.globus.org

Home+ File System

Scenario

- Need to host users home directories with access to multiple computational systems. Also need to host applications, compilers and temporary batch system files.

Questions

- What is the average file size?
- How often does data change?
- What applications need to be hosted?
  - How often do they change?
  - How often are they used? How many concurrent accesses?
- Are backups required?
- What data management policies need to be implemented?
- Is access required from compute nodes or only front-end nodes?
- What network interfaces are available?

Home+ File System

NCAR Solution

- DDN SFA7700 w/ mixed SSD and HDD
- Metadata and /usr/local application tree is pinned to SSD
- Home directories are on HDD with data replication
  - User quotas are enabled
  - Snapshots are enabled
  - Backups are enabled
- Batch system temporary files are contained in a ‘fileset’ with a quota
- System is connected to both Ethernet and IB networks
- File system is accessible from both front-end nodes and compute nodes
- Configured with a 1M block size (32K sub-block size)
- Configured with a 4K inode size (small files will actually reside in the inode)

Centralized Parallel File System

Scenario

- Different resources are available for different uses such as model runs, pre/post-processing, data analysis and visualization. As data has become larger, it becomes more difficult to move the data to the computational resource you need to use. Is there are way to have access to the same storage from all these different resources?

Questions

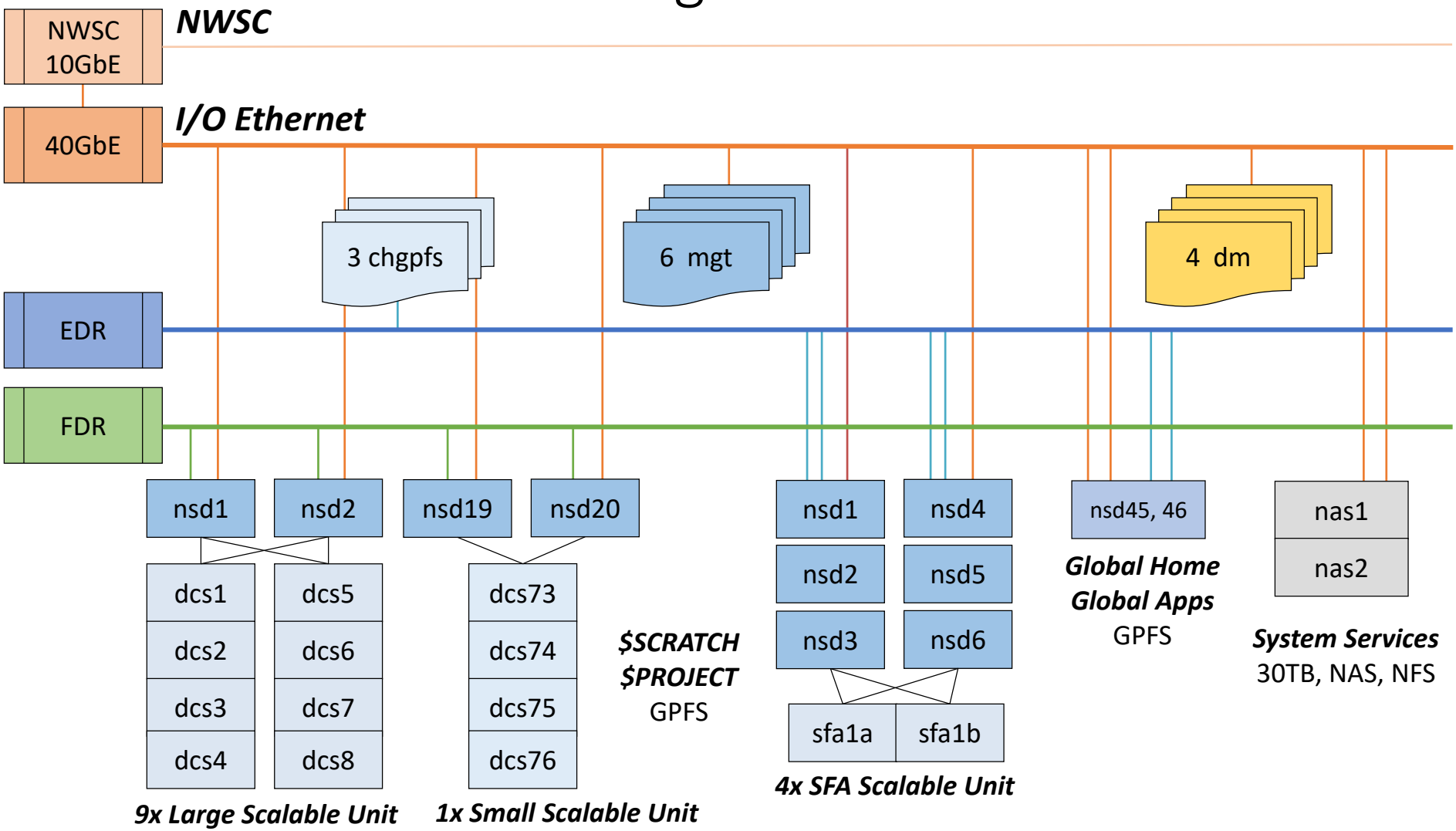
- What types of usage patterns are expected?
  - scratch space – data turns over fairly frequently
  - project space – groups working together on common problem with common data
  - visualization – need access to large data sets
- What is the average file size? How often does data change?
- What data management policies need to be implemented?
  - quotas, purge policies, allocation policies
- What network interfaces are available?
- Will I/O traffic be carried on the same network as computational traffic?
- How many client nodes need access?
- What are the bandwidth requirements for data access?
  - these may be different for each resource

Centralized Parallel File System

NCAR Solution

- Tier 1 – IBM DCS3700 storage systems with NL-SAS drives
  - Metadata is mixed with data on HDD
- Tier 2 – DDN SFA14KXE storage system with SSD and NL-SAS drives
  - Metadata resides on SSD
  - Data resides on HDD
- File Systems supported
  - scratch – 8M block size (256K sub-block size), 4K inode
  - project – 4M block size (128K sub-block size)
  - quotas enabled, no backups
  - 'filesets' are used to provide project containers
  - IB connected systems will have the best performance
  - Ethernet connected systems will have lower bandwidth available

NCAR Storage Architecture



IBM Spectrum Scale

NCAR GLADE Storage

LCI workshop on high performance clustered computing

Tier 1

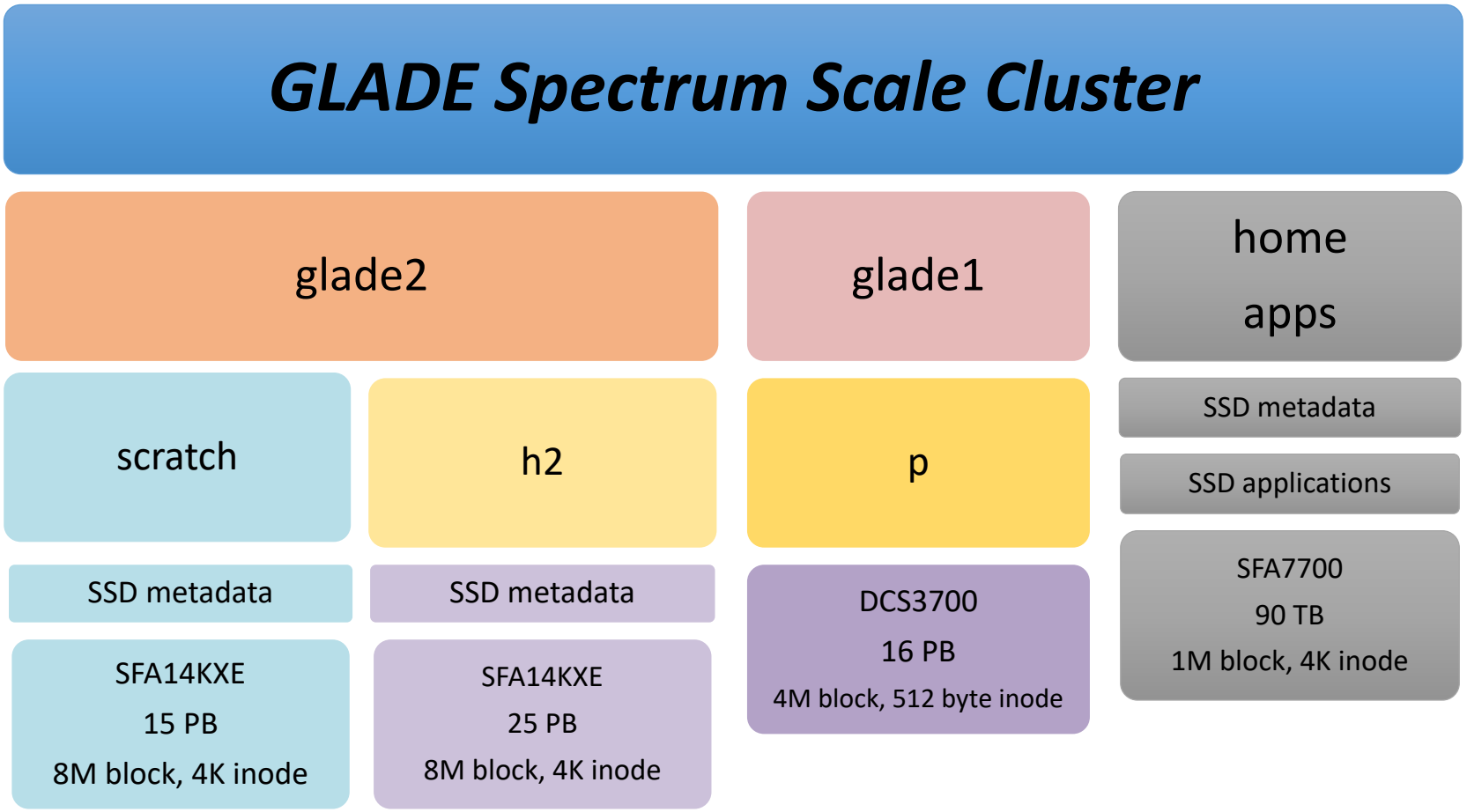
- *90 GB/s bandwidth*
- *16 PB useable capacity*
- *76 IBM DCS3700*
- *6840 3TB drives*
  - *shared data + metadata*
- *20 NSD servers*
- *6 management nodes*
- *File Services*
  - *FDR*
  - *10 GbE*

Tier 2

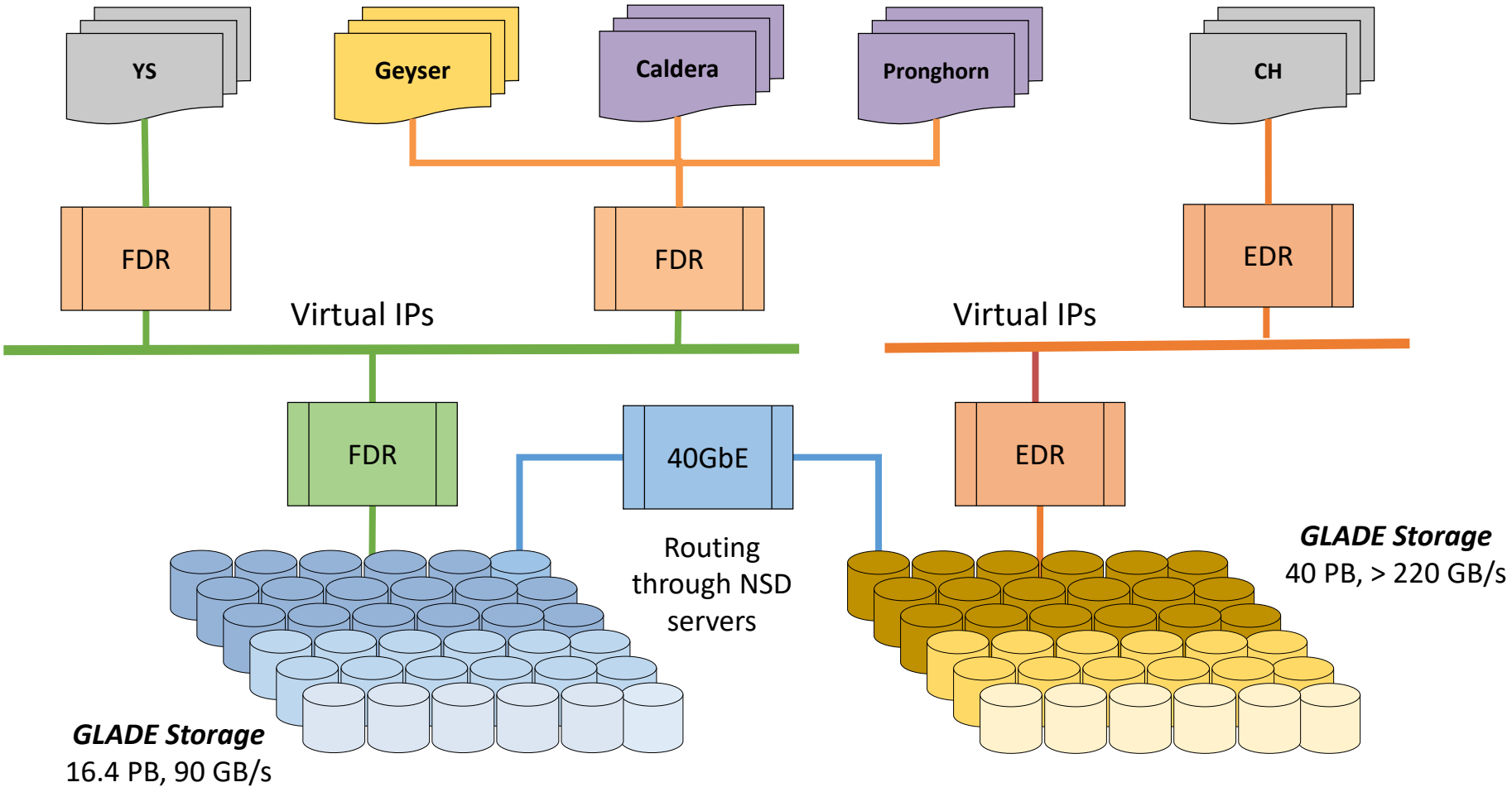
- *220 GB/s bandwidth*
- *40 PB useable capacity*
- *8 DDN SFA14KXE*
  - *32 NSD server VMs*
- *6560 8TB drives*
  - *data only*
- *96 800GB SSD*
  - *Metadata*
- *File Services*
  - *EDR*
  - *40 GbE*

DDN® STORAGE

NCAR GLADE File System



GLADE Routing



Best Practices, Hints & Tips

General Best Practices

- Benchmark the system whenever you can
  - Benchmark the system when you first get
    - This is your best case after your tuning and should be considered your baseline
  - Run your baseline again after any change to make sure performance hasn't been affected
  - Use general purpose network and storage performance benchmarks
  - Use benchmarks based on your user workloads if possible
- During installation and checkout, make sure to test all reliability/redundancy features of the system
  - Pull a drive while running
  - Fail a component in a failover pair
- Monitor your system
  - Look for issues prior to them becoming big issues
- Get feedback at regular intervals from your users
  - Are they having a good experience with you system or do they have pain points
  - How users ***feel*** about the system is more important than benchmark results

Performance Hierarchy From Best to Worst

1. Large records in sequential order
   a. Defined as larger than the file system block size
2. Large records in strided order or small records in sequential order
3. Large records in random order or small records in strided order
4. Issue hints when reading in small records in random order
   a. If supported by the file system
5. Small record random order without hints
   a. Defined as smaller than the block or sub-block size of the file system

Read:Write Balance

- Try to understand the balance of reads and writes across your system
- General rule of thumb in CS textbooks
  - Read: 90%
  - Write: 10%
- This doesn't tend to apply to HPC applications.
- For example, the ratio for many scientific applications
  - Read: 60% to 70%
  - Write: 30% to 40%
- HPC applications tend to each be unique
- This will affect your storage balance in the attempt to meet all your application requirements

Response Time vs. Bandwidth

Optimized for Response Time

- Tend to have larger variance
- Tend to complete *less* work per unit time
- Are best suited for:
  - Online applications
  - Interactive work loads
  - Real-time applications

Optimized for Bandwidth

- Tend to have smaller variance
- Tend to complete more work per unit time
- Are best suited for:
  - Batch work loads
  - Parallel applications

Avoid "Gold Plating"

- Rule of thumb
  - Configure a file system to handle peak performance up to 3 or 4 standard deviations above the mean to avoid "gold platting" (John Watts, IBM)
  - Programmers worried about performance will often over architect a system
  - Balance cost versus percentage of workload that actually need top performance
- When sizing the disk I/O subsystem
  - Don't ask about peak loads
  - Do ask
    - What are the highest mean loads?
    - How often do you see a peak load?
    - Do you have critical loads that justify additional cost?

Credit: Ray Paden, Parallel File System Guru

Mixing Workloads in a File System

Home Directory Usage

- Tends to have many small files
- Tends to support interactive or online work loads
- Large variance in access patterns, data rates and times of usage
  - E.g. used a lot during business hours and quiet during the night
- Stat() calls are frequent
  - E.g., ls –l
  - High IOPs access pattern
  - Tends to thrash metadata
- Rate of change is small
  - E.g., many files remain untouched for long periods of time

Scratch Directory Usage

- Principle working directory for applications runs
- Access tends to be during a batch job run
- 24x7 usage with consistent access patterns and data rates
- Stat() calls should be less frequent
- Rate of cahnge can be large
  - Data migrated to other storage tier's or tape soon after job run

Best Practice

- Avoid mixing home and scratch directories
- However, there are ways to do this if you really need to

Solutions For Mixed File Sizes

Example: Seagate ClusterStor Nytro Storage Systems

- Combine SSD and HDD in the storage solution
- Small I/O writes are sent to SSD storage
- Large I/O writes are sent directly to HDD storage
- Read's are satisfied from either SSD or HDD depending on where data currently resides
  - No read ahead to pull data to SSD first
- As SSD storage fills, oldest data is moved to HDD as a background task
- Applications can see up to 12x I/O speed benefit for small I/O [1]

[1] Seagate

Benchmarking

Benchmarking Tools

- Common Benchmarks
  - Good general benchmarks for HPC sized systems
    - For I/O testing (IOR, mdtest, xdd)
      - Spectrum Scale (gpfsperf)
    - For network testing (iperf3, ib_read_bw, ib_write_bw)
  - Good general benchmarks for smaller sized systems
    - For I/O testing (iozone, bonnie++)
  - Application benchmarks
    - Instrumented user applications
  - Benchmark Kernels
    - Spectrum Scale (nsdperf)
- Measurement Tools
  - OS tools (dd, iostat, nmon)
  - Storage System Tools
    - Unique to each storage system

Benchmarking Goals

- Baseline performance measurements
- Located slow performing components
  - Disks, LUNs
  - Network adapters
  - Network switches
- Performance Tuning
  - Test different file system parameters
    - Block sizes
    - RAID group sizes
    - Stripe sizes
  - Network tuning
    - send/receive queue sizes
    - IPoIB
    - MTUs

iperf3

https://github.com/esnet/iperf
fasterdata.es.net/performancetesting

- The iperf3 package of tools can be used to measure both TCP and UDP performance
- Good test to make sure an Ethernet network is performing as expected
  - Can help identify underperforming NICs or network switch ports
- Runs in a client/server mode
  - Will need to script to run tests between multiple nodes
  - Make sure to kill off your servers when you are done

Example

```
Startup Servers on two ports
iperf3 –s –D –p 5201
iperf3 –s –D –p 5202

Start tests between nodes
iper3f -f g -t 5 –c nsd21 -p 5201 & iperf3 -f g -t $5 –c nsd21 -p 5202
```

Ib_read_bw, ib_write_bw

https://community.mellanox.com/docs/DOC-2802

- Part of the Mellanox ***perftest*** package distributed with MOFED
  - Package contains bandwidth and latency benchmarks for both InfiniBand and Ethernet
- Good test for InfiniBand network health
  - Can help identify underperforming HBAs or switch port issues
- Runs in a client/server mode
  - Will need to script to run tests between multiple nodes
  - Make sure to kill off your servers when you are done

Example

```
On Server
ssh nsd21; ib_read_bw -d mlx5_0 -i 1 -R -z –report_gbits

On Client
ssh  nsd22; ib_read_bw -d mlx5_0 -i 1 -R -z -D 2 nsd21 --report_gbits
```

IOR HPC Benchmark

sourceforge.net/directory/system-adminstration/benchmark/?q=IOR

- Specifically design for parallel I/O benchmarking
- Utilizes MPI for parallel I/O
- Can be run from the command line, through a script or as a batch job
- Can run different types of I/O
  - POSIX, MPIIO, HDF5 or NCMPI
- Can run with multiple tasks
- Can configure block size and transfer size to use
- Has specific options for Lustre file systems
- Many, many more options

Example

```
ior –w –r –o <filename>
```

mdtest

sourceforge.net/projects/mdtest

- mdtest is an MPI-coordinated metadata benchmark test that performs open/stat/close operations on files and directories and then reports the performance. [1]
- Uses MPI for communications
- Creates multiple files per process
  - Can specify a shared file or file per process
  - Can specify that all files go into a single directory or into unique directories
- Can run in POSIX and MPIIO modes

Example

```
mpiexec –npernode 32 –n 2 –hostfile <hostfile> mdtest
–d /scratch/mdtest –z 0 – b 0 –n 1024 –C –T –r -F
```

[1] sourceforge

dd

- Never run as root, you will clobber your file system!!
- Reads from an input file “if” and writes to an output file “of”
- You don’t need to use real files
  - Can read from devices e.g. /dev/zero, /dev/random, etc.
  - Can write to /dev/null
- Caching can be misleading
  - Use direct I/O (oflag=direct)

Example

```
$ time dd if=/dev/zero of=./test.dd bs=1G count=1 oflag=direct
1+0 records in
1+0 records out
1073741824 bytes (1.1 GB) copied, 37.8403 s, 28.4 MB/s
```

iozone

- Common test utility for read/write performance
  - Tests storage “peak” aggregate performance
- Can test both single-node and multi-node configurations
  - In multi-node, the network can become a significant contributor to performance measurement
- Sensitive to caching, use “-I” for direct I/O.
- Can run multithreaded (-t)
- Simple, ‘auto’ mode:

  iozone –a
- Or pick tests using ‘-i’:

  -i 0: Read/re-read

  -i 1: write/rewrite

Example

```
iozone -i 0 -i 1 -+n –e -r 128k –s <file_size> -t <num_threads> -+m <hostfile>
```

Bonnie++

- Comprehensive set of tests
  - Create files in sequential order
  - Stat files in sequential order
  - Delete files in sequential order
  - Create files in random order
  - Stat files in random order
  - Delete files in random order (--wikipedia)
- Just ‘cd’ to the directory on the file system, then run ‘bonnie++’
- Uses 2x client memory (by default) to avoid caching effects
- Reports performance (K/sec, higher are better) and the CPU used to perform operations (%CP, lower are better)
- Highly configurable, check its man page!

Other Options

How About Cloud and Big Data

Design/Standards

- POSIX : Portable Operating System Interface (NFS, GPFS, Panasas, Lustre)
- REST: Representational State Transfer, designed for scalable web services

Case-specific solutions

- Software defined, hardware independent storage (e.g. Swift)
- Proprietary object storage (e.g. S3 for AWS, which is RESTful)
- Geo-replication: DDN WOS, Azure, Amazon S3
- Open Source object storage: Ceph vs Gluster vs …
- Big data (map/reduce): Hadoop Distributed File System (HDFS), QFS, …

LCI workshop
on high performance clustered computing

Characteristics

- Open source parallel object storage from RedHat
- Runs on a single distributed compute cluster
- Provides interfaces for object-level, block-level, and file-level storage
- Metadata services and storage are segregated from data services and storage
- Uses XFS as the underlying file system

System Components

- ceph-mon – monitors cluster for active and failed cluster nodes
- ceph-mds – stores metadata of inodes and directories
- ceph-osd – object storage devices, stores the file contents in an XFS file system
- ceph-rqw – REST gateways to expose objects

Sources

- Ceph – ceph.com
- Ceph Wiki – tracker.ceph.com/projects/ceph/wiki/

Gluster FS

LCI workshop
on high performance clustered computing

Characteristics

- Open source software defined storage by RedHat
- Runs on most hardware, flexible deployment options
- Runs across Ethernet of InfiniBand networks
- Utilizes a client/server model
- Implemented in user space, utilizes FUSE (File System in Userspace)
- No metadata server, metadata is stored with the file

System Components

- brick – basic unit of storage
- glusterd– cluster management daemon
- trusted storage pool – a trusted network of storage servers
- volume – a logical collection of bricks

Sources

- Gluster – www.gluster.org
- Gluster Docs – gluster,readthedocs.io/en/
- Red Hat Gluster Storage – www.redhat.com/en/technologies/storage/gluster

Characteristics

- Open source software framework for distributed storage from Apache
- Designed for use with the MapReduce programming model
- Utilizes node local storage
- Files are split into blocks and distributed to compute nodes
- Computation on the node uses the data it has locally
- Output is then pulled back together onto a centralized storage system
- Good for applications that can divide the problem into independent chunks like data processing tasks
- Not as good for interactive analytic or iterative tasks

System Components

- Hadoop Common – base framework
- Hadoop Distributed File System (HDFS) – node local file system
- Hadoop YARN – resource management system
- Hadoop MapReduce – Hadoop implementation of the MapReduce programming model

Sources

- hadoop.apache.org

Future Directions

Hybridization of storage

- Mixed media within a system
  - SSD, HDD
- Tier'ing storage with different capabilities
  - Fast tier, slow tier, archive tier
- Connecting different storage types
  - Seamless migration between storage solutions
  - Object store <-> Object store
  - POSIX <-> Object store
  - Local <-> Cloud

Burst Buffers (Yet Another Cache)

- Intel Cache Acceleration Software
- DDN Infinite Memory Engine
- IBM FlashCache

Ethernet connected drives

- Seagate's Kinetic interface
- HGST's open Ethernet drive

Wrap Up

Conclusions

- There is no one right solution for HPC Storage systems
- Each solution is highly dependent on your environment
- The more time you spend gathering requirements and talking with your stakeholders, the better your design
- Trade-offs are inevitable, focus on prioritizing basic features
- Benchmarking can be a valuable long-term tool
- Technology changes frequently, but the basic principles are fairly stable
- Reach out to the community, most are more than willing to share experiences
- Good Luck!!!!

Other Resources

- Spectrum Scale User Group
  - www.spectrumscale.org
- Spectrum Scale Wiki
  - www.ibm.com/developerworks/community/wikis/home
  - Search for Spectrum Scale
- OpenSFS - Lustre
  - opensfs.org
- Lustre Wiki
  - wiki.lustre.org
- Enterprise Storage Forum
  - www.enterprisestorageforum.com
- OrangeFS
  - www.orangefs.org
- MarFS
  - github.com/mar-file-system/marfs
- ESNet
  - fasterdata.es.net

Acknowledgments

- The materials for these courses evolve as technology and practices change over time. Materials are updated with each new workshop to reflect these changes. We would like to acknowledge previous instructors who's work lays the foundation for each new workshop.
- Previous Instructors
  - Mehmet Belgin – Georgia Tech University
  - Wesley Emeneker – Georgia Tech University
  - Alex Younts – Purdue University

Thanks for Participating!

pjg@ucar.edu