A Survey of HPC Storage Systems

ATPESC 2023 - Track 7 - I/O

August 10, 2023

Michael J. Brim, R&D Staff

National Center for Computational Sciences (NCCS) /
Oak Ridge Leadership Computing Facility (OLCF)

Oak Ridge National Laboratory

Overview and Goals for this Lecture

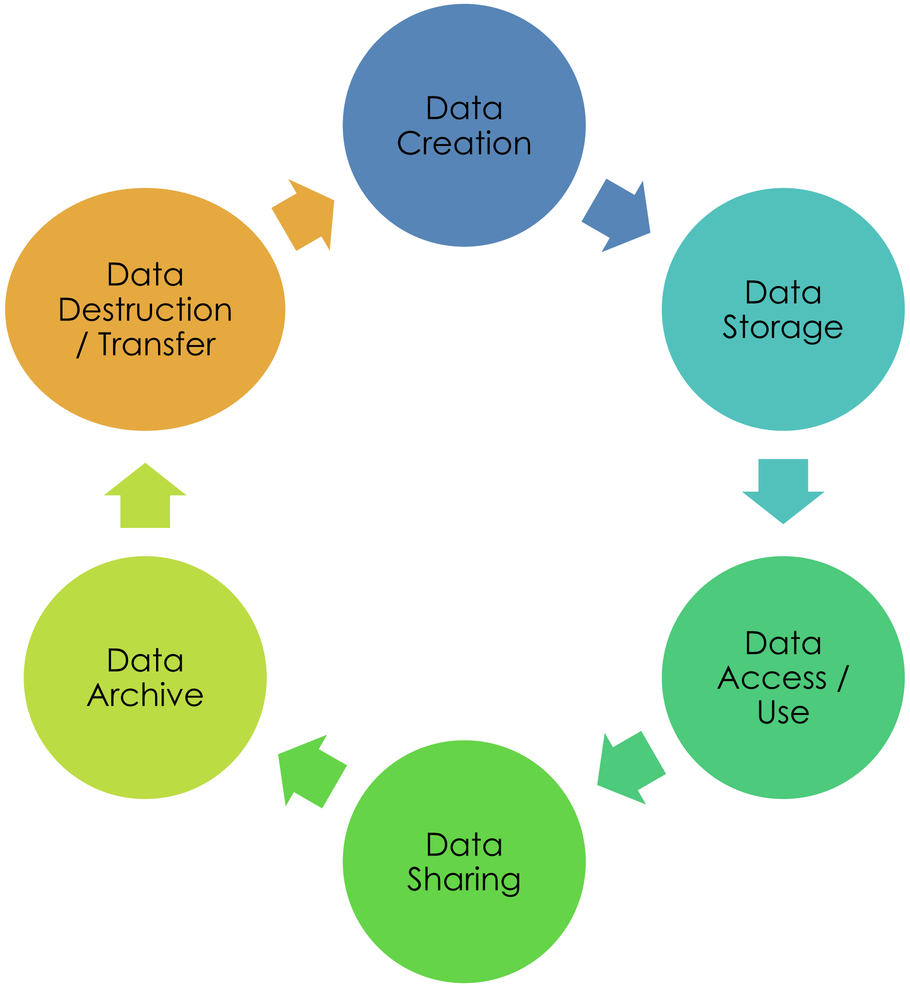
1. Introduce the various types of HPC storage systems and their intended use cases
2. Survey widely-used HPC storage systems and their architectures
3. Discuss how HPC storage is evolving and potential implications for users

Oak Ridge National Laboratory

Types of HPC Storage

- Data Lifecycle
- HPC Storage Use Cases
- Storage Classification Terminology

DOE Computing Facilities Support the Full Data Lifecycle



- Creation and Access characteristics often dictate the type of Storage system on which the data is stored
- Important characteristics: (not exhaustive)
  1. Total data size
  2. Size of data in working set (% of total size)
  3. Number of processes that write/update the data
  4. Number of processes that read the data
  5. I/O access patterns, concurrency and frequency
     - *producer/consumer*: readers wait until writers are done
     - *bulk-synchronous I/O*: distinct write phases, readers see data from last completed write phase
     - *free-for-all*: no coordination between accesses from different processes (**Advice: Don't do this!**)

HPC Storage has many distinct use cases

- "Software": Facility-managed system & user software (e.g., applications, libraries, system configuration)
- "Home": Per-user or per-project storage for application code and data, job scripts, documents, etc.
- "Scratch": High-performance storage for runtime use by jobs
- "Archive": Long-term data storage for archival & sharing

- Each use case has different requirements, in terms of:
  - storage space; data security and sharing; data lifetime; I/O access pattern, concurrency and performance

HPC Storage Requirements by Use Case

Use Case	Storage Space	Storage Lifetime	Data Sharing	I/O Throughput	I/O Latency	Access Concurrency
Software	Small/Medium (10s TB)	Long-term (System Lifetime)	Yes	Low (MB/sec)	Medium	Yes (reads)
User Home	Medium (100s TB)	Medium-term (Months)	No	Medium (GB/sec)	Medium	Possibly (reads)
Project Home	Medium (100s TB)	Medium-term (Months)	Yes	Medium (GB/sec)	Medium	Likely (reads)
Scratch	Large (10s/100s PB)	Short-term (Days/Weeks)	Yes	High (TB/sec)	Low	Yes (writes and reads)
Archive	Very Large (100s PB)	Long-term (Years)	Yes	Medium (GB/sec)	High	Not Likely

Storage System Classification: Terminology

- Private vs. Shared
- Local vs. Remote
- Centralized vs. Distributed
- Serial vs. Parallel
- Single-tier vs. Multi-tier

Storage System Terminology: Private vs. Shared

- Private: storage is used by a single actor
- Shared: storage is used by many actors
- Point-of-View is important!
  - Actors may be users, compute hosts, processes, jobs, etc.
  - You can have a storage system that is private to a job, but shared amongst processes in the job

Storage System Terminology : Local vs. Remote

- Local: access to storage uses only local data paths
  - the definition of ”local data paths” can be a bit fuzzy in new HPC system architectures
  - historically, it meant “local to the host where the process performing the access is located”

- Remote: access to storage is via the network

Storage System Terminology : Centralized vs. Distributed

- Centralized: a client interacts with a single storage server
- Distributed: a client may interact with many storage servers

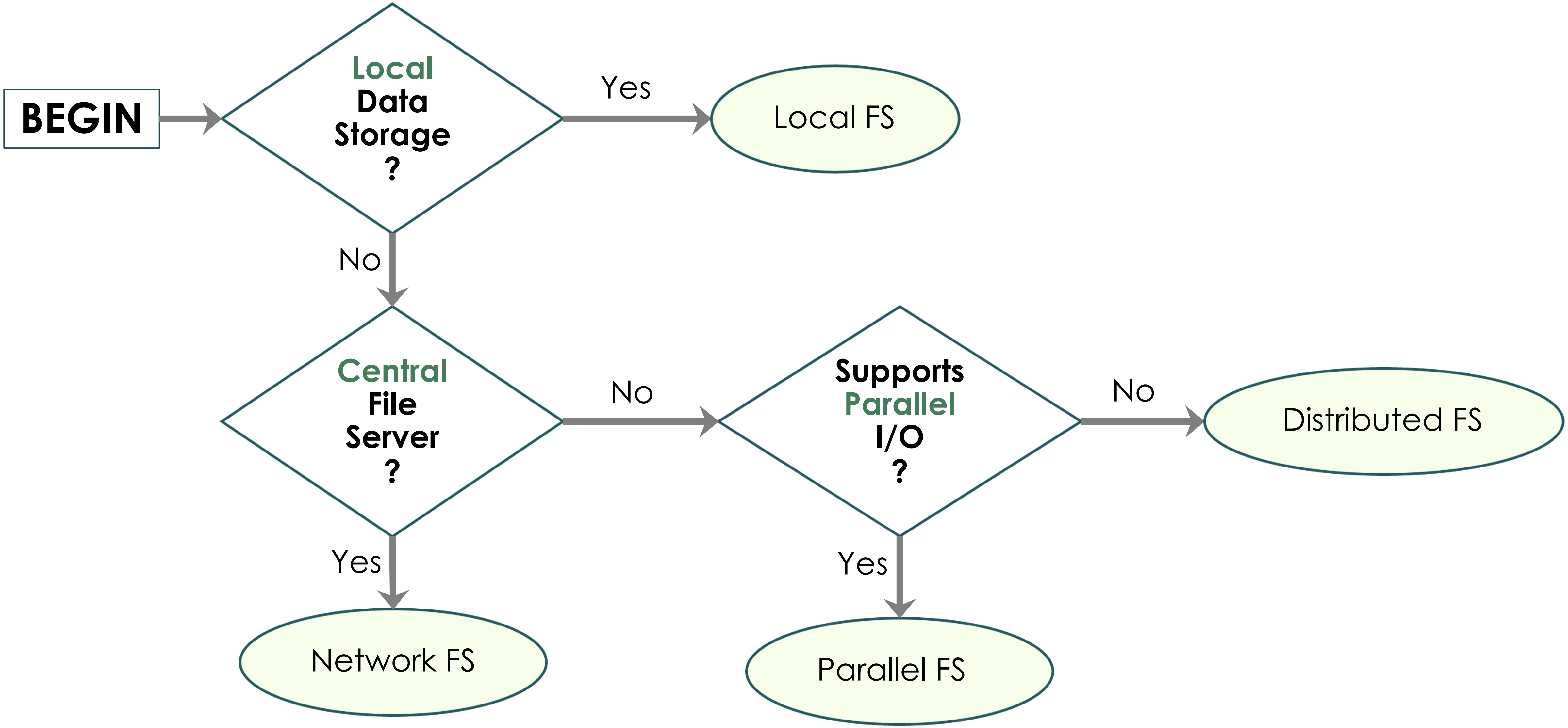
Storage System Terminology : Serial vs. Parallel

- Serial: a single client accesses a particular data item (e.g., a file) on the storage at any given time
  - serial accesses can be made by separate clients, they just don't overlap in time
- Parallel: many clients access a particular data item at the same time

Storage System Terminology : Single-Tier vs. Multi-Tier

- Single-Tier:
  - Physical: a storage system includes one layer of storage devices
  - Logical: clients interact with a single storage system
- Multi-Tier:
  - Physical: a storage system includes multiple layers of storage devices
  - Logical: clients have access to two or more storage systems

Storage Terminology Example: Shared File Systems



Oak Ridge National Laboratory

Survey of HPC Storage Systems by Use Case

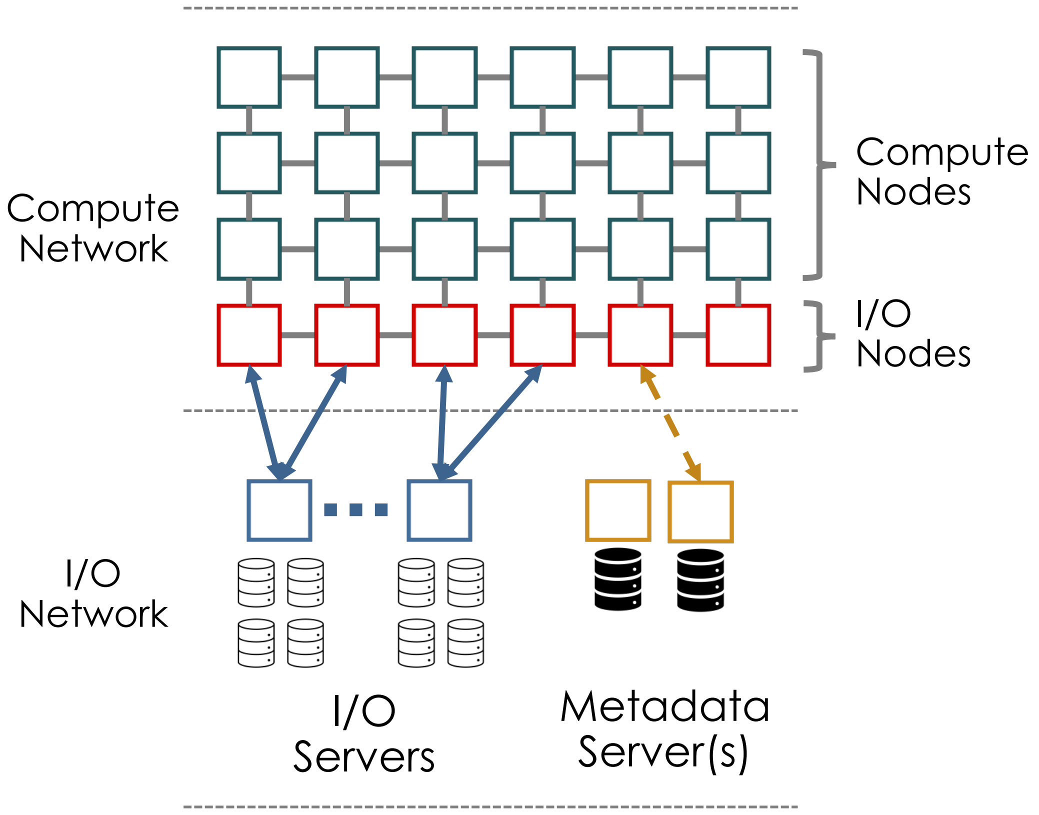
- Scratch Storage
- Home Storage
- Software Storage
- Archive Storage

Scratch Storage Systems

- Purpose: High-performance, short-term storage for runtime use by HPC jobs
- Common Solutions
  – Parallel File Systems (PFS)
  – Burst Buffers (BB)
  – Node-local Storage (NLS)

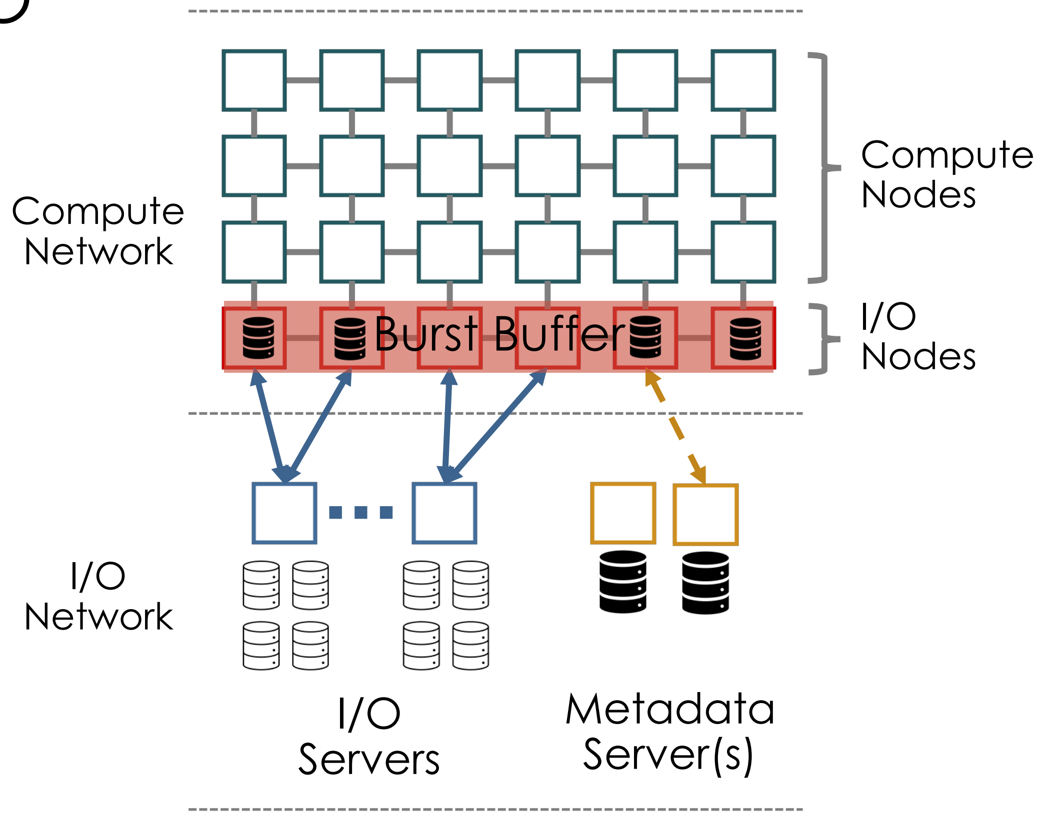
Scratch Storage - Parallel File Systems

- Goal: Enable high-performance I/O for a large number of concurrent clients
- Production Examples: Lustre, IBM Spectrum Scale (GPFS), Panasas PanFS
- Key Architectural Features
  - separate FS metadata and I/O servers
    - "do one thing well"
  - spread file data across I/O servers
    - helps to load balance I/O traffic
  - clients directly access I/O servers in parallel
    - helps to maximize I/O bandwidth



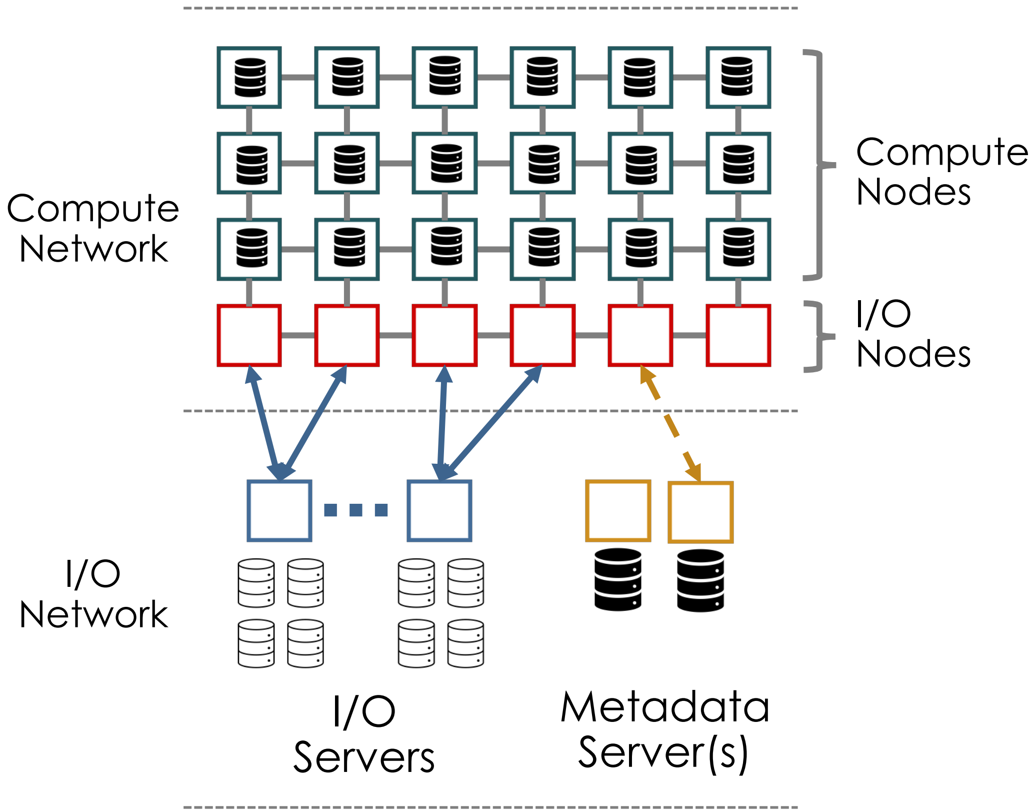
Scratch Storage - Burst Buffers

- Goal: Reduce load on a parallel file system to enable faster application I/O
  - by buffering bursty writes (e.g., simulation outputs or checkpoints)
  - by caching data from "hot" reads
- Production Examples: Cray DataWarp
- Key Architectural Features
  - "Faster" storage (e.g., Flash SSD) closer to Compute
  - Automatic data staging to/from PFS
  - Reservation-oriented (e.g., resources dedicated to specific jobs)



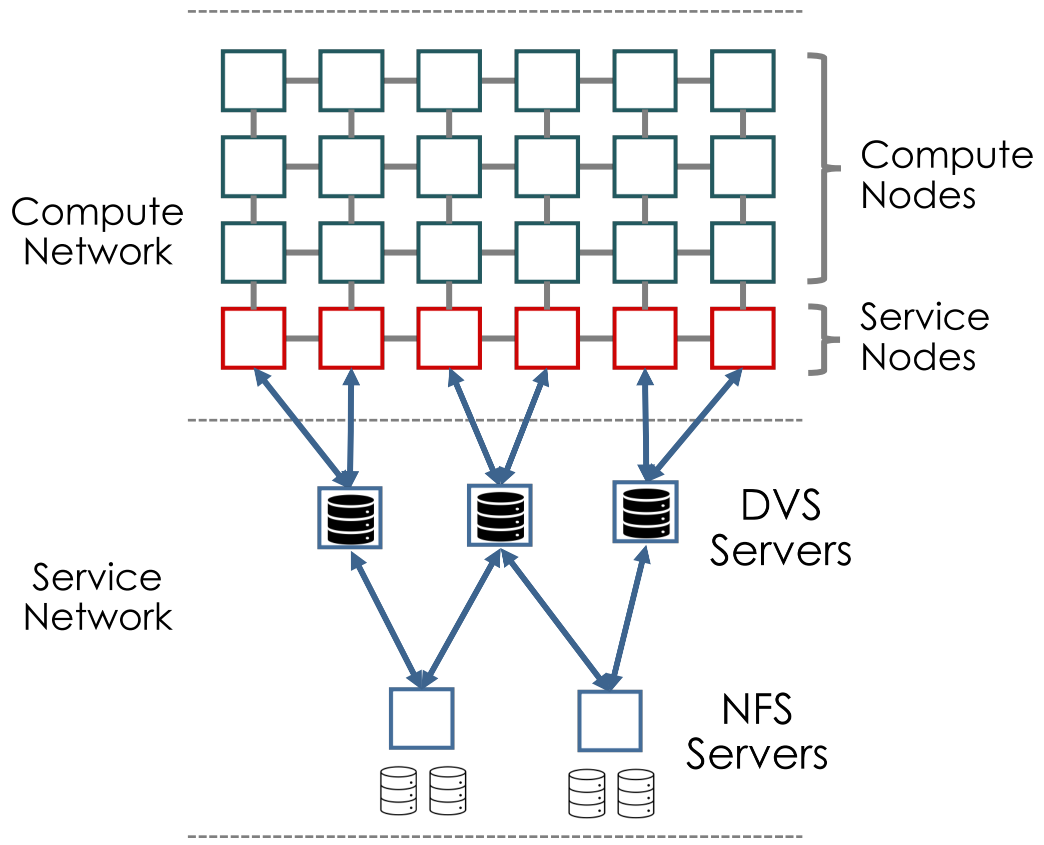
Scratch Storage - Node-local Storage

- Goal: Reduce load on a parallel file system to enable faster application I/O
  - by buffering bursty writes (e.g., simulation outputs or checkpoints)
  - by pre-staging data for "hot" reads
- Production Examples: NVMe SSDs in compute nodes
- Key Architectural Features
  - "Fastest" storage co-located with Compute
  - Storage capability scales linearly with allocated job nodes
  - However, software solutions required for:
    - runtime sharing of data in NLS across nodes (e.g., UnifyFS)
    - moving data to/from PFS



Home Storage Systems

- Purpose: User or Project storage
- Common Solutions
  - Network File Systems
    - Production Examples: NFSv4
  - Distributed File Systems
    - Production Examples: HPE/Cray Data Virtualization Service (DVS)
  - Parallel File Systems
    - Production Examples: Lustre, GPFS



Software Storage Systems

- Purpose: Facility-managed system & user software
- Common Solutions
  - Local File System
    - Production Examples: sys/app software cache on NLS
  - Network File Systems
    - Production Examples: NFSv4
  - Distributed File Systems
    - Production Examples: HPE/Cray DVS
  - Parallel File Systems
    - Production Examples: Lustre, GPFS

Archive Storage Systems

- Purpose: Long-term data storage for archival & sharing
- Common Solutions
  - Tape Libraries
  - High Performance Storage System (HPSS)
    - software optimized for efficient and performant use of tape libraries
    - also supports classes of different storage media and hierarchical tiering

Survey Summary: Storage @ DOE Computing Facilities

Center	Home	Software	Scratch	Archive
ALCF	Lustre	Lustre	Lustre, NLS	HPSS
NERSC	GPFS[1]	GPFS[1]	Lustre	HPSS
OLCF	NFS	NFS, DVS[2], NLS	GPFS[1], Lustre, NLS	HPSS

[1]NOTE: IBM Spectrum Scale is the product name for GPFS.

[2]NOTE: DVS is the Cray Data Virtualization Service.

HPC Storage is Evolving

- HPC Storage Challenges
- Requirements Driving Evolution
- HPC Storage Architecture Trends

Challenge: Exascale Systems have Arrived

- For 30+ years, DOE HPC meant scalable modeling and simulation
  - bulk-synchronous checkpoint/restart (C/R) was the primary I/O requirement (write-dominated, mostly sequential accesses)
    - PFS are designed to do C/R well, while still providing POSIX I/O semantics
- C/R for full-scale applications on exascale systems is problematic
  - expected system component failure rates require more frequent checkpoints for application progress
  - number of potential I/O clients exceeds the capabilities of most PFS

Challenge: Data-intensive Science is Widespread

- Data-intensive Science is pushing the current limits of HPC Storage
- Large-scale data analysis (e.g., ML model training) is read-dominated, and often uses repeated random accesses from varying processes
- Experimental data from instruments with very large data generation rates (e.g., LHC, SKA) is currently difficult/impossible to store in lossless form

HPC Storage Evolution - New Requirements

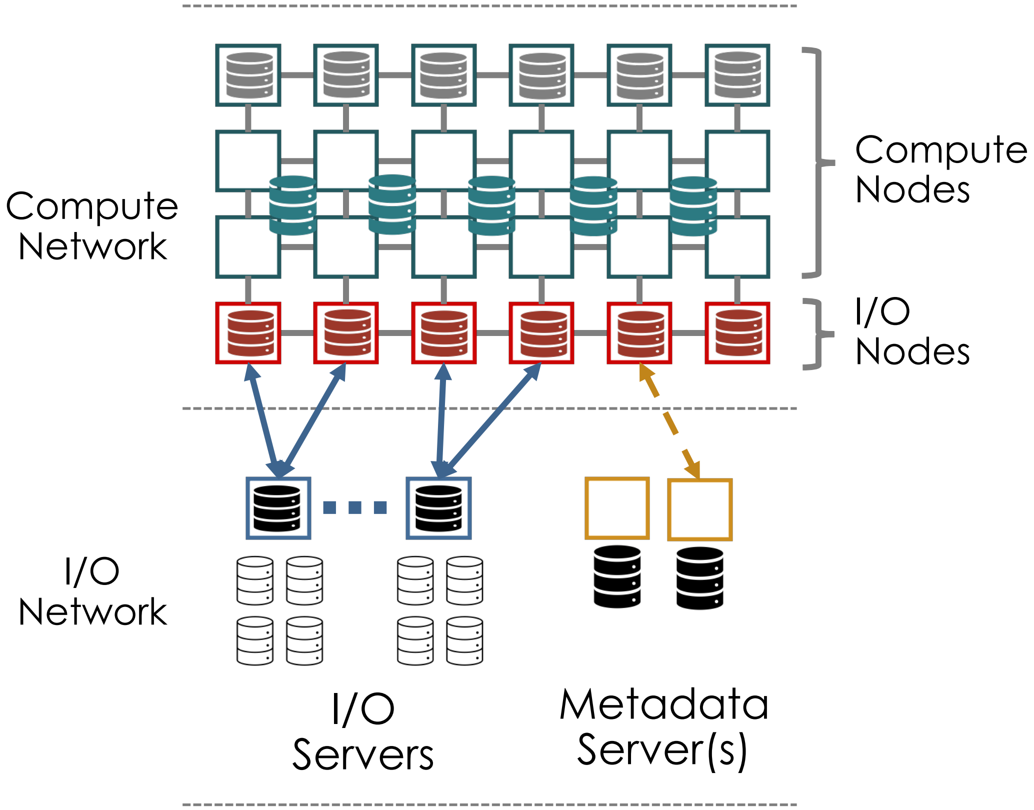
- Interfaces and Access Patterns
  - Reads are as important as writes (maybe more important)
  - POSIX file read/write is rarely the right I/O semantic for scalable HPC workloads
    - many workloads may benefit from alternatives such as:
      - simple put/get of data objects, possibly with object versioning
      - publish/subscribe
      - streaming data

HPC Storage Evolution - New Requirements

- Storage advances from the cloud may be beneficial to HPC
  - and are frequently integral to deployment of popular data analysis frameworks on HPC systems
- Cloud Data Abstractions and Interfaces
  - Key-value and columnar data stores are better suited for many data analysis workloads involving queries
  - Graph analysis benefits from custom storage
  - Analysis of real-time streaming data from many sources
- Cloud Storage Technologies
  - Elastic provisioning of storage resources
  - Quality-of-service (QoS) or service-level agreements (SLA)

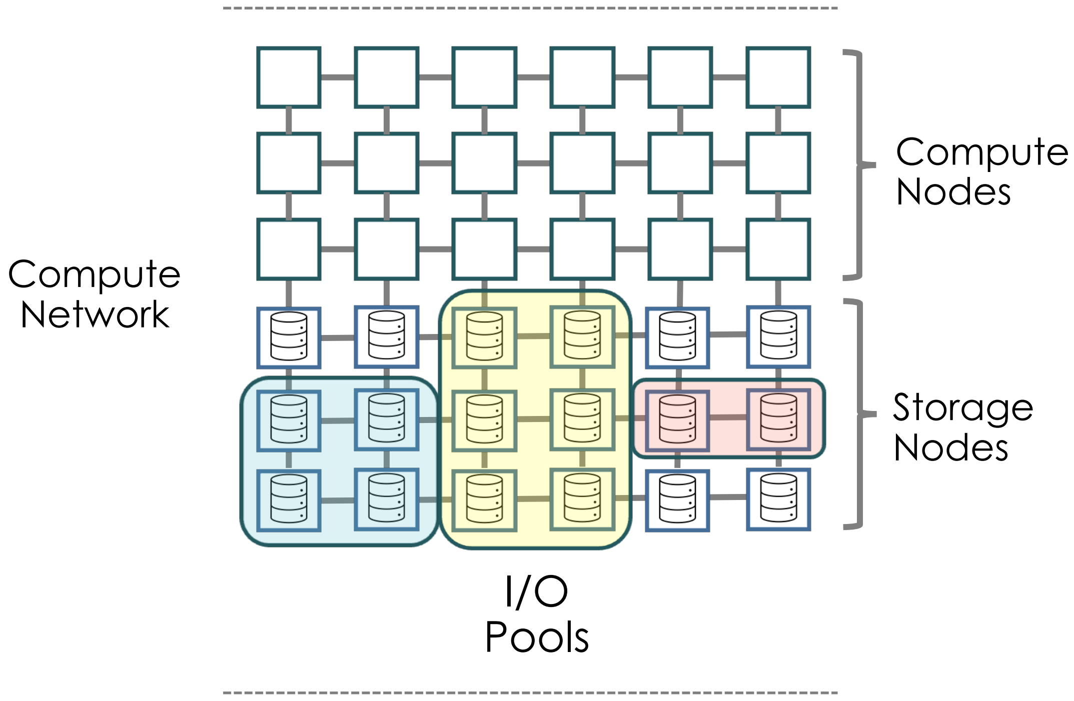
Storage Architecture Trends - Tiering for Performance

- More use of "fast" non-volatile memory-based storage devices
  - including hardware optimized for key-value or put/get semantics
- More storage tiers
  - within HPC PFS and Archive (mostly transparent to users)
  - between HPC Compute and PFS (multi-tier burst buffers)
  - within HPC Compute (node-local storage, near-node-local storage)



Storage Architecture Trends - Storage Disaggregation

- Storage disaggregation
- Allocate network-attached storage resources dynamically to pools
  - pools provide raw I/O on blocks or objects, not POSIX
- Assign pools to jobs
- Pools may also provide storage QoS guarantees



HPC Storage User Advice

"This all sounds very confusing. How do you expect users to deal with such complexity?"

- HPC I/O libraries provide higher-level abstractions for managing and accessing scientific data (e.g., hierarchical groups of datasets)
  - HPC storage experts are busily trying to hide all this complexity under the covers of the existing I/O libraries
- Vendors are also developing new storage abstractions and interfaces that help manage the complexity (e.g., DAOS)

Discussion/Questions

brimmj@ornl.gov

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725