PARALLEL FILE SYSTEMS

Filbert Minj

Storage Team @SERC,

Indian Institute of Science

filbert@iisc.ac.in

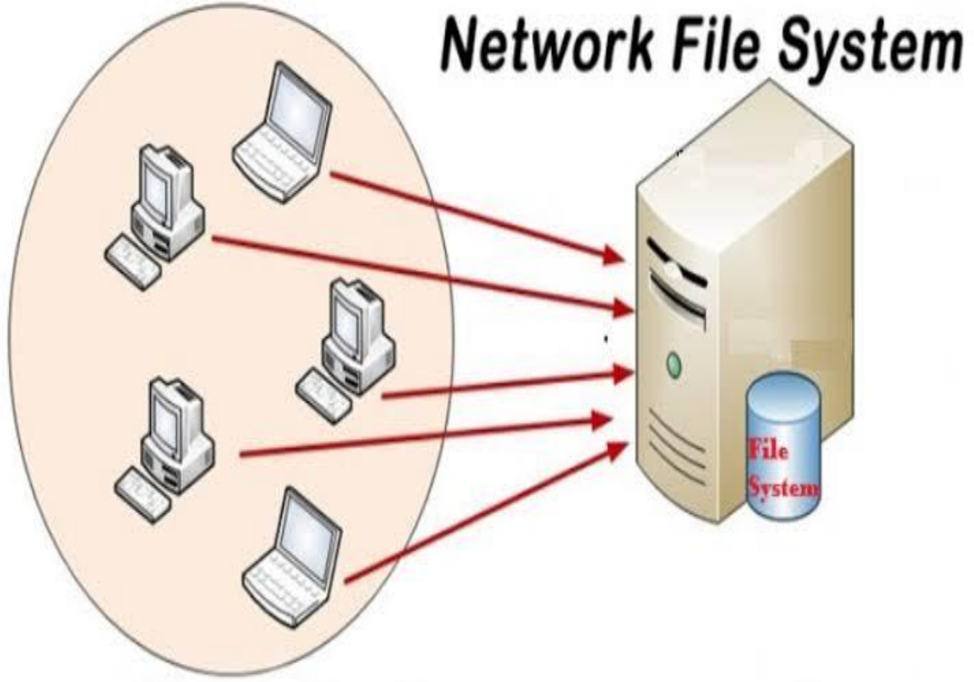
OUTLINE OF THE CONTENTS

FILE SYSTEMS

- File Systems have two key roll
  - Organizing and maintaining the named space
    - Directory hierarchy and file names that let us find things
  - Storing contents of files
    - Providing an interface through which we can read and write data
- Local file systems are used by a single operating system instance (client) with direct access to the disk
  - E.g NTFS, ext4 on laptop
- Distributed file systems provide access to one or more clients who might not have direct access to the disk
  - e.g. NFS, AFS, etc.

DISTRIBUTED FILE SYSTEM (DFS)

- Distributed File System (DFS) is a method of storing and accessing files based in a client/server architecture
- In a distributed file system, one or more central servers store files that can be accessed, with proper authorization rights, by any number of remote clients in the network
- Example: Network File System (NFS)
- Distributed file systems can be used by parallel programs, but they have significant disadvantages:
  - The network bandwidth of the server system is a limiting factor on performance
  - To retain UNIX-style file consistency, the DFS software must implement some form of locking which has significant performance implications



PARALLEL FILE SYSTEMS

- Store application data persistently
  - usually extremely large datasets that can’t fit in memory
- Provide global shared namespace (files, directories)
- Designed for parallelism
  - concurrent (often coordinated) access from many clients
- Designed for high-performance
  - operate over high-speed networks (IB, Myrinet, Portals)
  - optimized I/O path for maximum bandwidth

COMMON USE CASES OF PARALLEL FILE SYSTEMS

- Parallel file systems historically have targeted high-performance computing (HPC) environments that require access to large files, massive quantities of data or simultaneous access from multiple compute servers
- Applications include climate modeling, computer-aided engineering, exploratory data analysis, financial modeling, genomic sequencing, machine learning and artificial intelligence, seismic processing, video editing and visual effects rendering

EXAMPLES OF PARALLEL FILE SYSTEMS

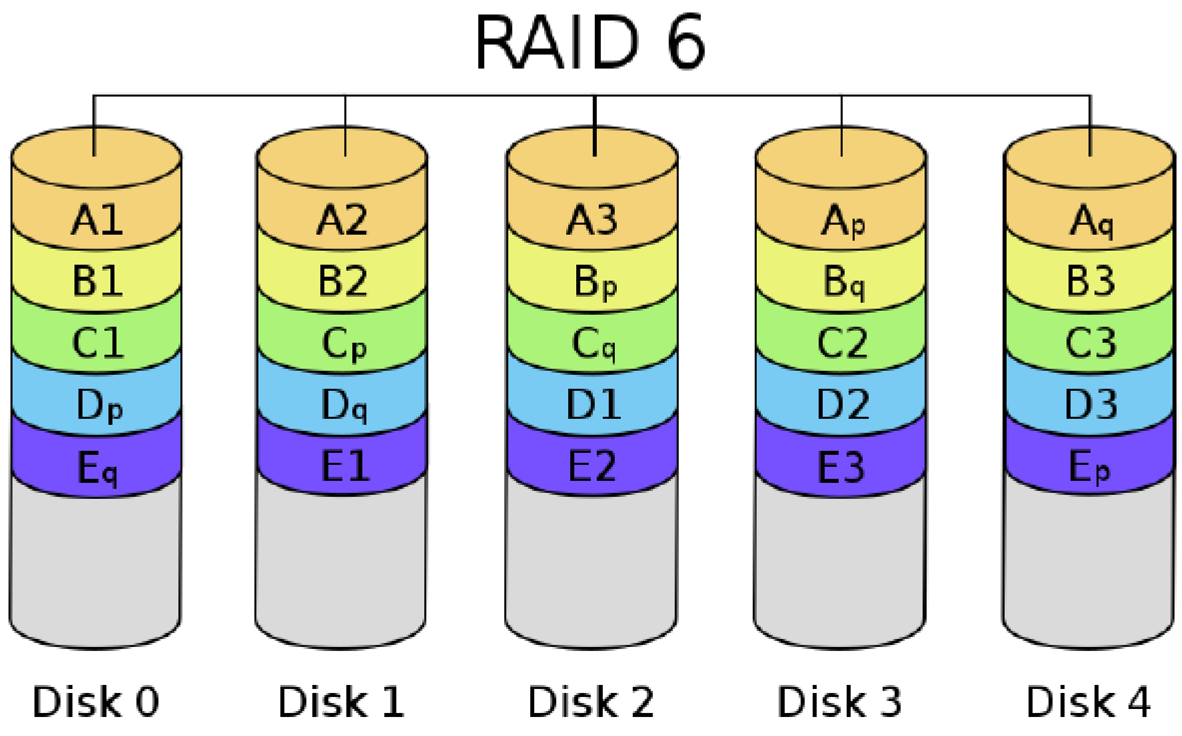
- General Parallel File System (GPFS) / IBM Spectrum Scale
  - Developed by IBM
  - Available for AIX and Linux
- Lustre
  - Developed by Cluster File Systems, Inc. (bought by Sun)
  - Movement towards **OpenLustre**
  - Name is amalgam of Linux and clusters
- Parallel Virtual File System (PVFS)
  - Platform for I/O research and production file system for cluster of workstations
  - Developed by Clemson University and Argonne National Laboratory

STORAGE DEVICE

- Single hard drive
  - File system resides entirely on a single disk
- RAID (Redundant Array of Independent Disks)
  - A logical disk built of many physical disks
  - A stripe of data is stored across multiple disks
  - Each chunk is placed on a single disk
  - Several different levels of RAID with different protection and performance characteristics
  - RAID-6 (8+2) is typically used for distributed, parallel storage

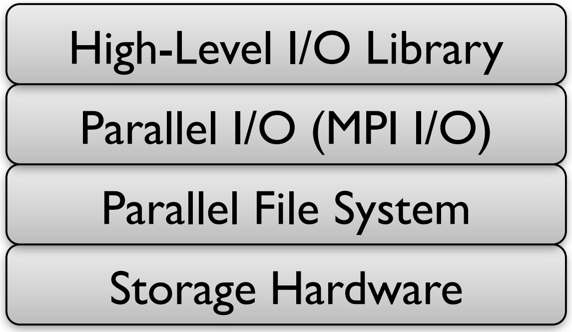
STORAGE DEVICE

- RAID-6 (N+M)
  - Erasure encoding allows up to M devices to fail without data loss
  - Trade off capacity/performance with data protection
  - Diagram is a 3+2 RAID-6 configuration



CHARACTERISTICS OF PARALLEL FILE SYSTEMS

- Three Key Characteristics:
  - Various hardware I/O data storage resources
  - Multiple connections between these hardware devices and compute resources
  - High-performance, concurrent access to these I/O resources
- Multiple physical I/O devices and paths ensure sufficient bandwidth for the high performance desired
- Parallel I/O systems include both the hardware and number of layers of software



PARALLEL I/O TECHNIQUES – MOTIVATION

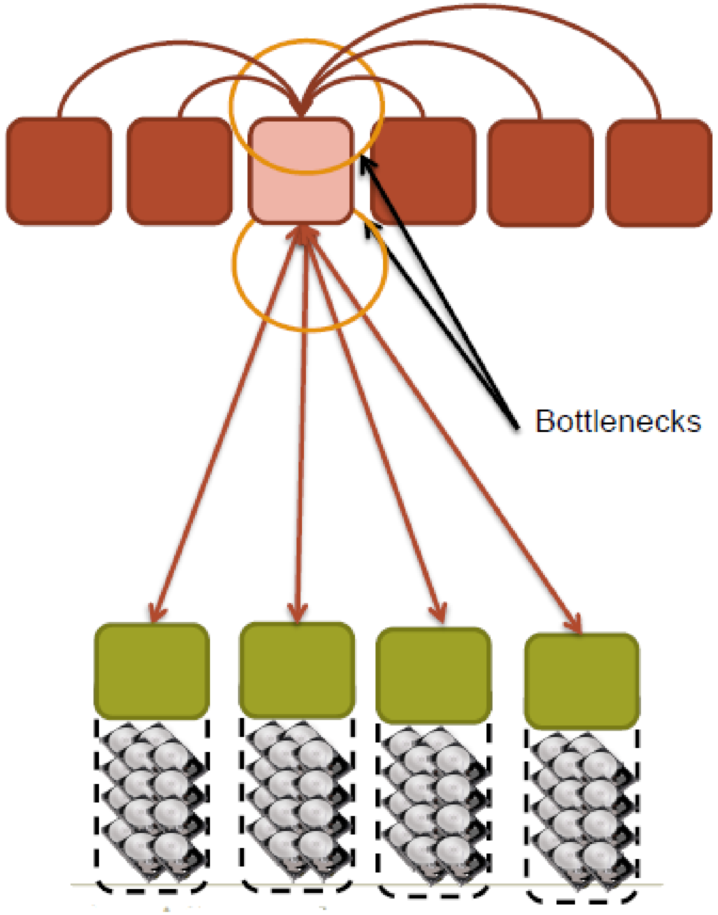
- Parallel applications that emphasize on the importance of data
  - Not all data-intensive or data-driven applications are ‘big data’ (volume)
  - HPC simulations of the real world that generates very large volumes of data
- Synthesize new information from data that is maintained in distributed (partly unique) repositories and archives
  - Distributed across different organizations and computers/storages
- Data analysis applications that are ‘I/O bound’
  - I/O dominates the overall execution time
  - I/O performance crucial for overall performance

WHAT MEANS I/O?

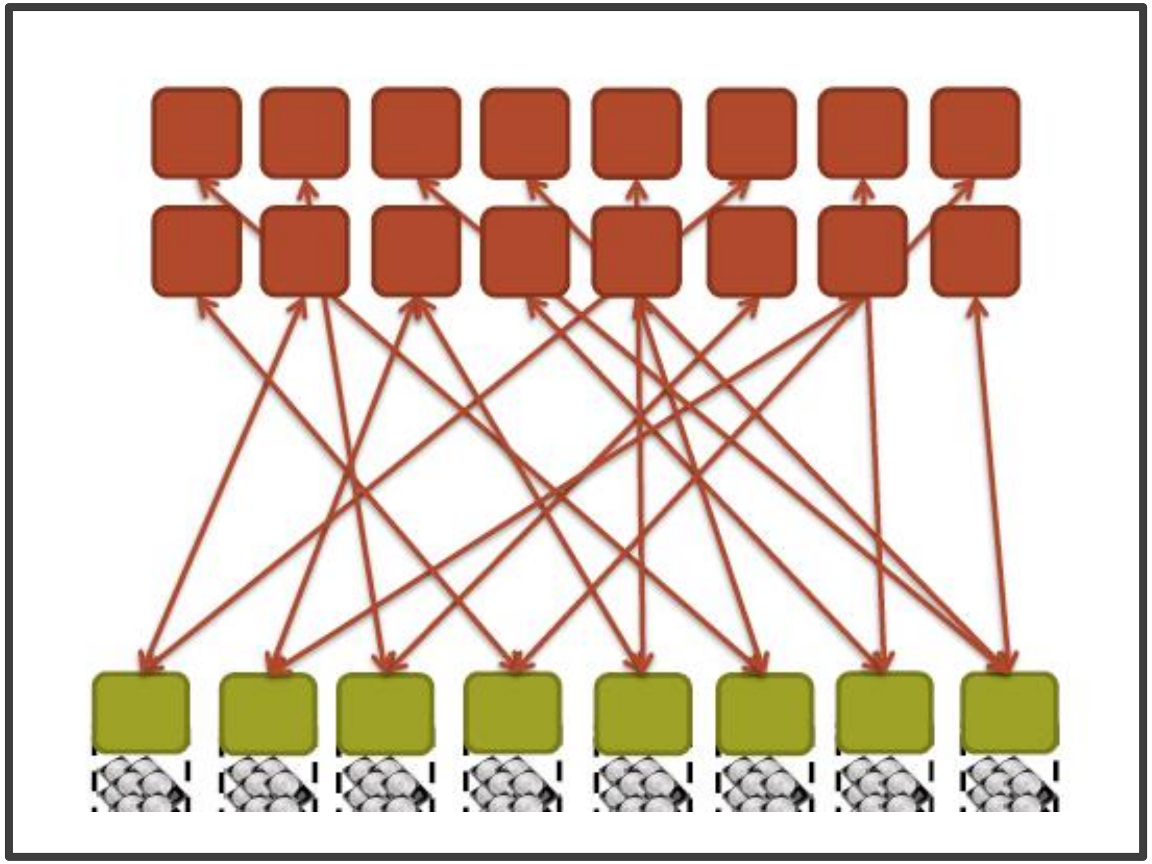
- Input/Output(I/O) stands for data transfer/migration from memory to disk (or vice versa)
- Important (time-sensitive) factors within HPC environments
  - Characteristics of the computational system (e.g. dedicated I/O nodes)
  - Characteristics of the underlying filesystem (e.g. parallel file systems, etc.)

I/O STRATEGIES: SPOKESPERSON (SEQUENTIAL I/O)

- One process performs I/O
  - Data Aggregation or Duplication
  - Limited by single I/O process
- Easy to program
- Pattern does not scale
  - Time increases linearly with amount of data
  - Time increases with number of processes
- Care has to be taken when doing the all-to-one kind of communication at scale
- Can be used for a dedicated I/O Server



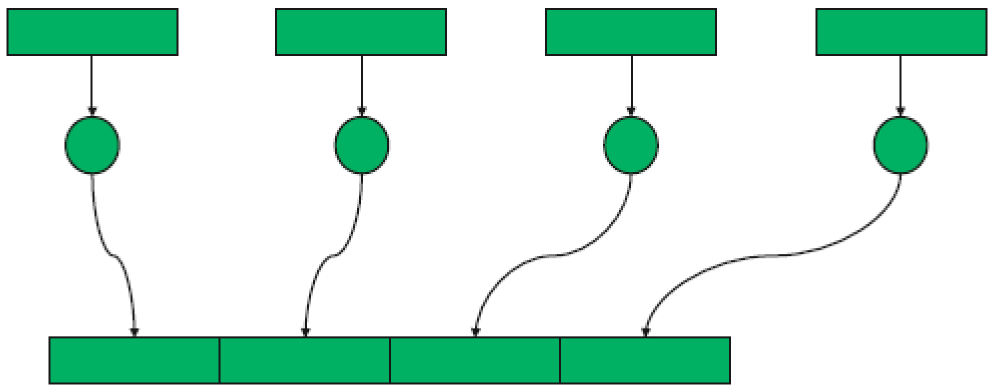
I/O STRATEGIES: MULTIPLE WRITERS – MULTIPLE FILES



- All processes perform I/O to individual files
- Easy to program
- Pattern may not scale at large process counts
  - Number of files creates bottleneck with metadata operations
  - Number of simultaneous disk accesses creates contention for file system resources
  - Hard to read back from diff number of processes

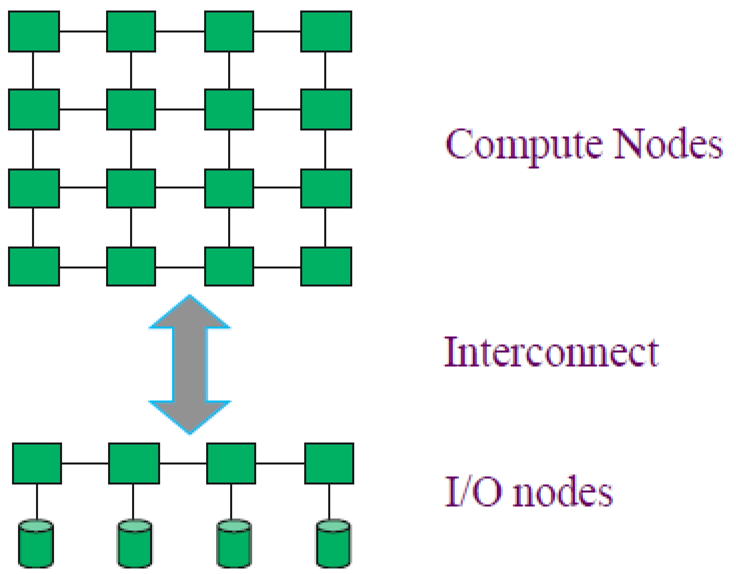
WHAT IS PARALLEL I/O?

- From user's perspective:
  - Multiple processes or threads of a parallel program accessing data concurrently from a *common* file (shared)
- Results in a single file and we can get good performance



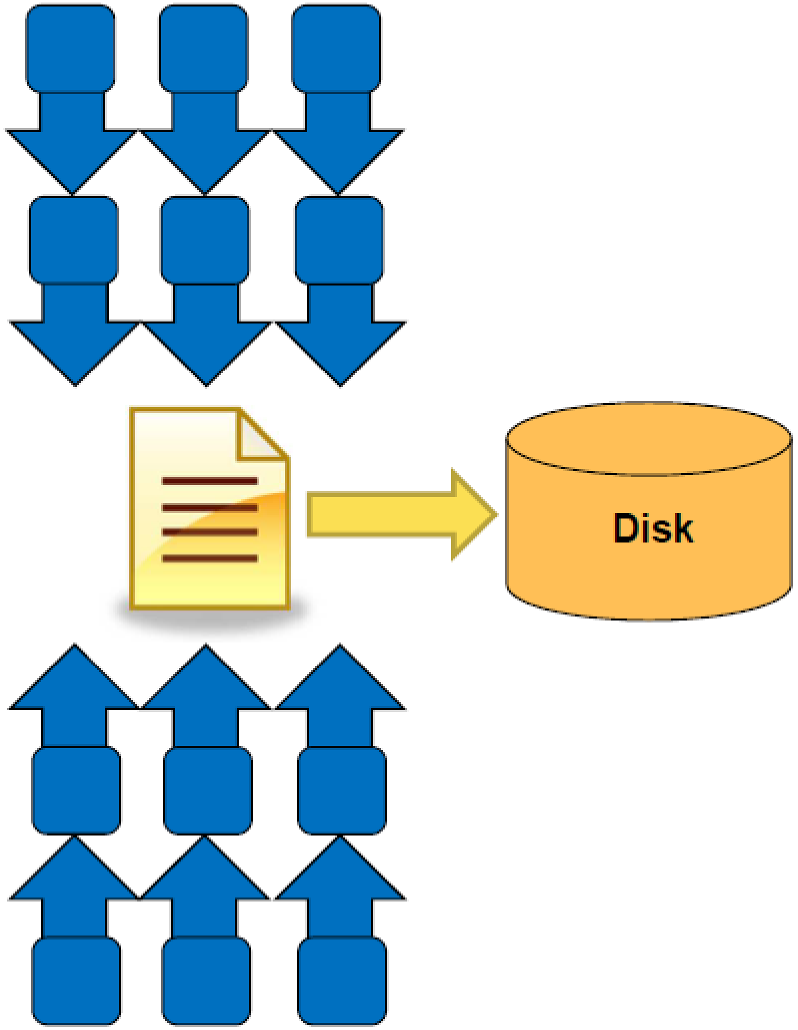
WHAT IS PARALLEL I/O? …

- From system perspective:
  - Files striped across multiple I/O servers
  - File system designed to perform well for concurrent writes and reads (parallel file system)

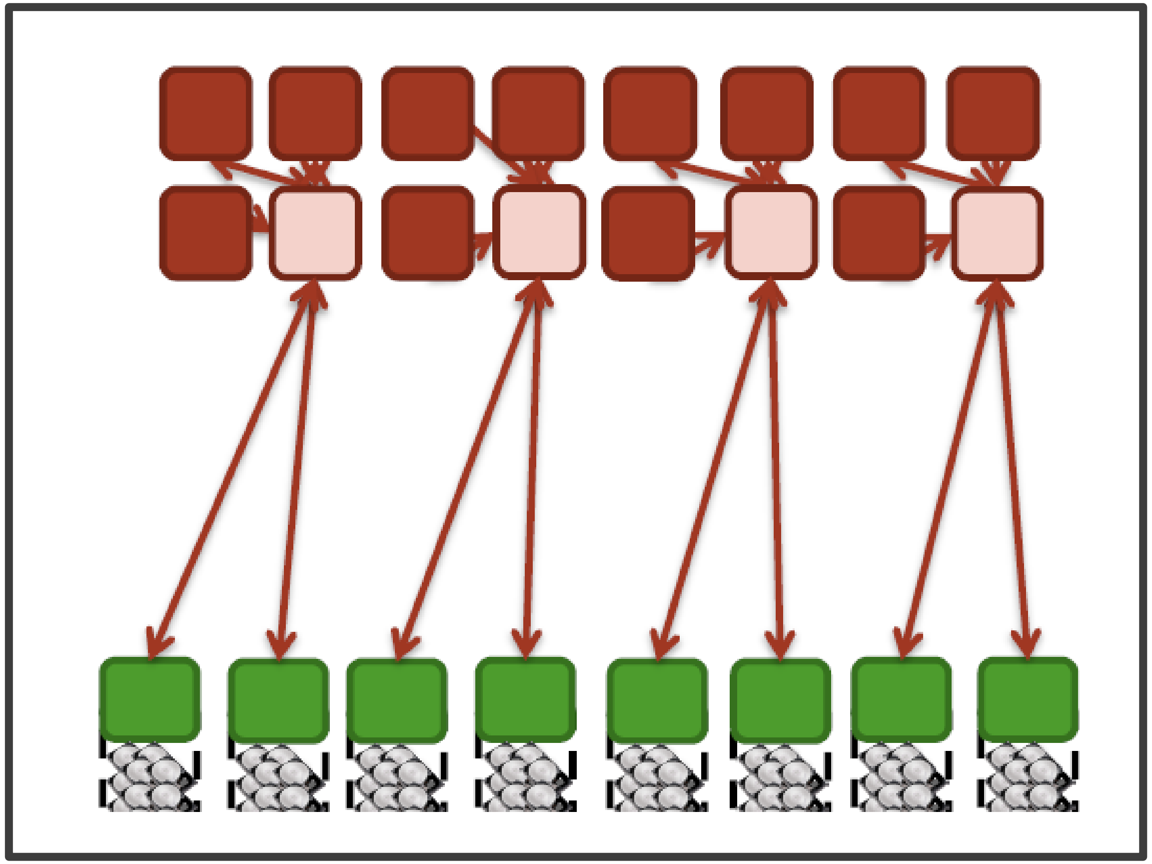


PARALLEL I/O: SHARED FILE

- Each process performs I/O to a single file which is shared
  - The file access is 'shared' across all processors involved
  - E.g. MPI/IO functions represent 'collective operations'
- Scalability and Performance
  - Data layout within the shared file is crucial to the performance
  - High number of processors can still create contention for file systems

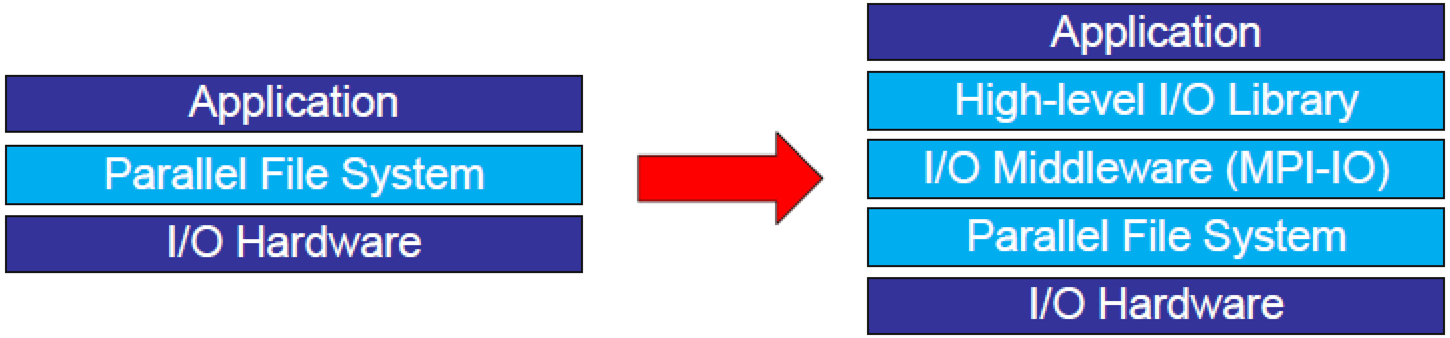


I/O STRATEGIES: COLLECTIVE IO TO SINGLE OR MULTIPLE FILES



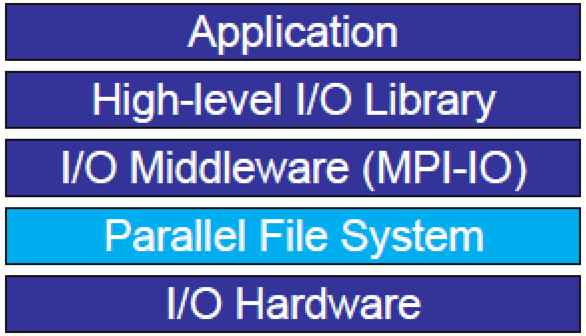
- Aggregation to a processor in a group which processes a subset of the total data
  - Serializes I/O in group
- Group of processes perform parallel I/O to a shared file
  - Decreases number of processes which access a shared file

PARALLEL I/O TOOLS FOR COMPUTATIONAL SCIENCE



- Application require more software than just a parallel file system
- Break up support into multiple layers with distinct roles:
  - Parallel file system (PFS) maintains logical space, provides efficient access to data (e.g. PVFS, GPFS, Lustre)
  - Middleware layer deals with organizing access by many processes (e.g. MPI-IO, UPC-IO)
  - High level I/O library maps app. abstractions to a structured, portable file format (e.g. HDF5, Parallel netCDF)

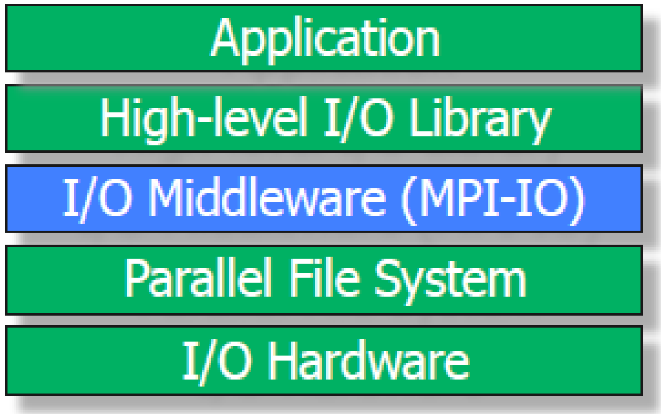
PARALLEL FILE SYSTEM



- Manage storage hardware
  - Present single view
  - Stripe files for performance
  - Focus on concurrent, independent access
  - Transparent : files accessed over the network can be treated the same as files on local disk by programs and users
  - Publish an interface that middleware can use effectively
  - Scalable

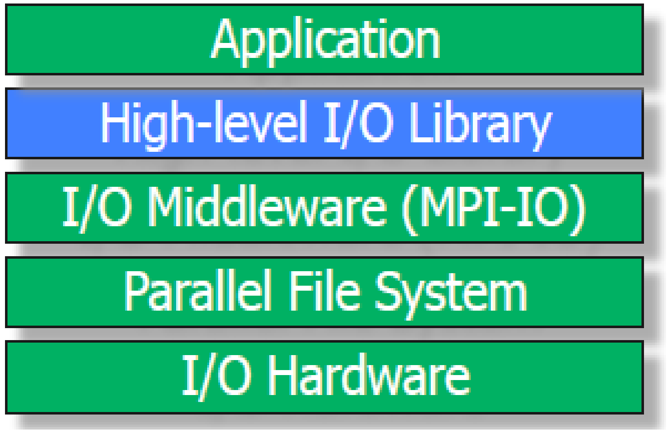
I/O MIDDLEWARE

- Facilitate concurrent access by groups of processes
  - Collective I/O
- Expose a generic interface
  - Good building block for high-level libraries
- Match the underlying programming model (e.g. MPI)
- Efficiently map middleware operations into PFS ones
  - Leverage any rich PFS access constructs

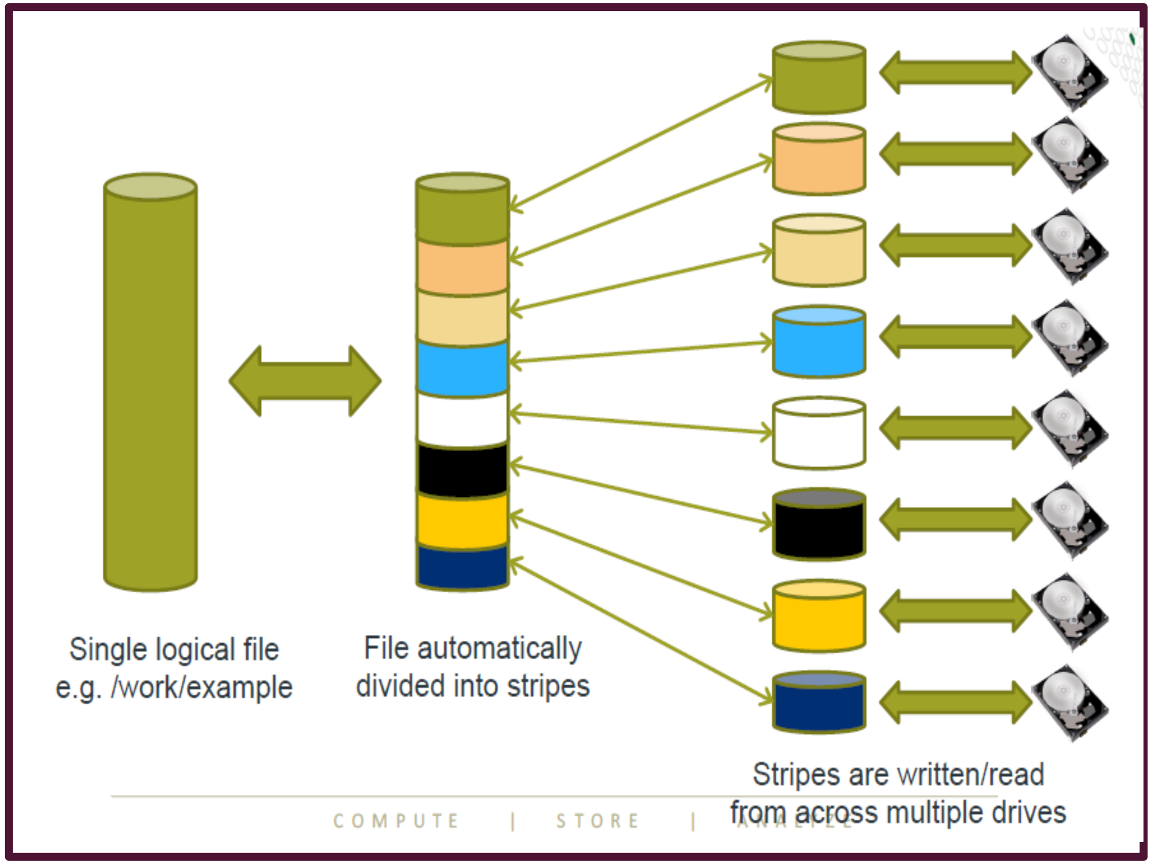


HIGH LEVEL LIBRARIES

- Examples: HDF-5, PnetCDF
- Provide an appropriate abstraction for domain
  - Multidimensional datasets
  - Typed variables
  - Attributes
- Self-describing, structured file format
- Map to middleware interface
  - Encourage collective I/O
- Provide optimizations that middleware cannot
  - e.g. caching attributes of variables



PARALLEL FILE SYSTEMS AND PERFORMANCE

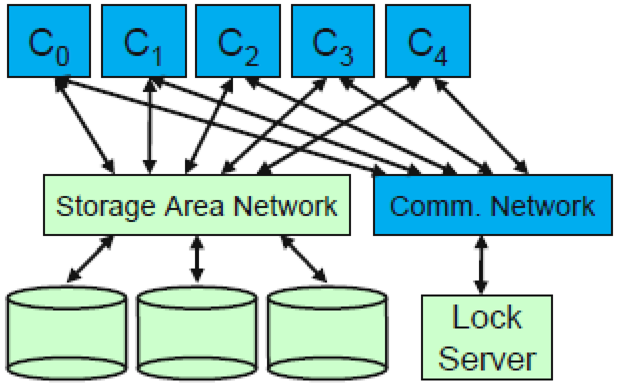


- Striping is the basic mechanism used in parallel file system to improve performance
  - Striping refers to a technique where one file is split into fixed-sized blocks that are written to separate disks in order to facilitate parallel access

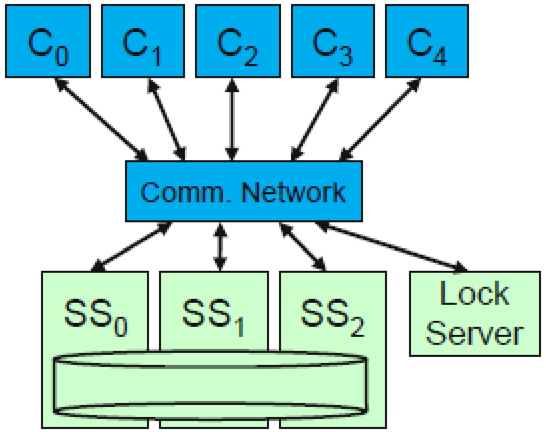
PARALLEL FILE SYSTEM ARCHITECTURES

- Two types of parallel file systems
- Shared Storage Architectures
  - Make blocks of disk array accessible by many clients
  - Clients operate on disk blocks
- Object Server Architectures
  - Distribute file data to multiple servers
  - Clients operate on regions of files or objects and Disk blocks are not visible to clients

SHARED STORAGE ARCHITECTURES



Shared storage using separate SAN

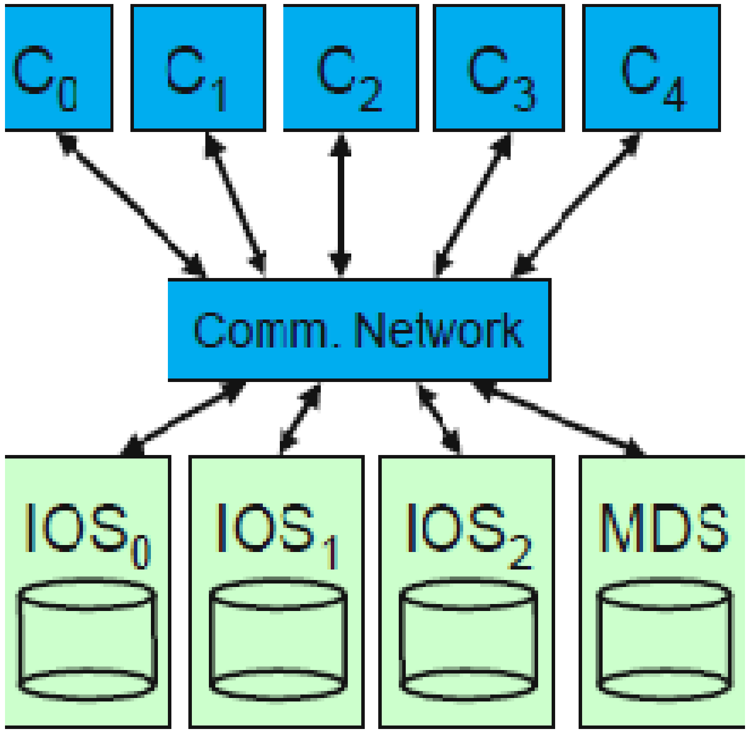


Pooled storage using existing interconnect

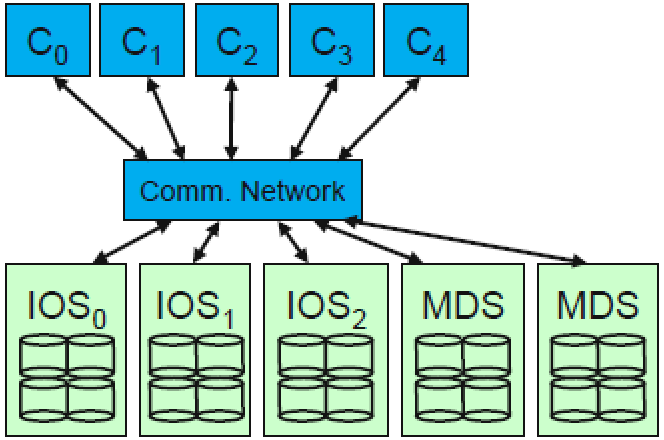
- Clients share access to disk blocks on real or virtual disks
  - Directly via Fibre-Channel SAN, iSCSI, AT over Ethernet
  - Indirectly via storage servers
    - e.g. Virtual Shared Disk, Network Shared Disk
    - May expose devices directly, or pool them into a larger whole
- Lock server coordinates shared access to blocks
  - May be a distributed service to reduce contention

OBJECT SERVER ARCHITECTURES

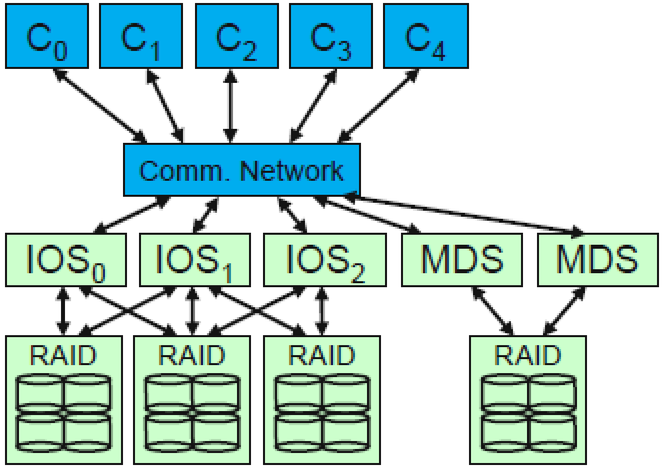
- Clients share access to files or objects
- Servers are "smart"
  - Understand something about the structure of data on storage
  - I/O servers (IOS) manage local storage allocation
    - Map client accesses into local storage operations
- Metadata server (MDS) stores directory and file metadata
  - Often a single metadata server stores all metadata for file system
- Locking is often required for consistency of data and metadata
  - Typically integrated into other servers
  - Atomic metadata operations can eliminate need for metadata locking



REDUNDANCY WITH OBJECT SERVER



Redundancy with replicated local storage



Redundant storage connectivity for failover

- Data may be stored on multiple servers for tolerance of server failure
  - Orchestrated either by client or servers
- Servers may have access to other server's data
  - Take over when a server fails
- In both cases, each server is primarily responsible only for its own data

LUSTRE FILE SYSTEM OVERVIEW

- An open-source distributed, parallel file system
- Three server roles
  - Metadata Server (MDS)
  - Object Storage Server (OSS)
  - Management Server (MGS)
- Client
  - No file system data locally accessible (excluding cache)
- Designed for scalability, high-performance, and high-availability
- Lustre runs on Linux-based operating systems and employs a client-server network architecture

LUSTRE COMPONENTS

- Management Server (MGS)
  - Communicates over a network
  - Provides services related to file system configuration information
  - Uses locally attached storage MGT (management service storage target) to store configuration data
  - /mnt/lustre (Lustre file system at SERC) has one MGS and one MGT

LUSTRE COMPONENTS …

- **Metadata Server (MDS)**
  - Communicates over a network
  - Provides services related to file system metadata such as directory contents, file names, attributes, and file layout
  - Uses locally attached storage to store metadata information
    - Metadata Target (MDT)
    - An MDS has one or more MDTs
  - /mnt/lustre has one MDS and one MDT:

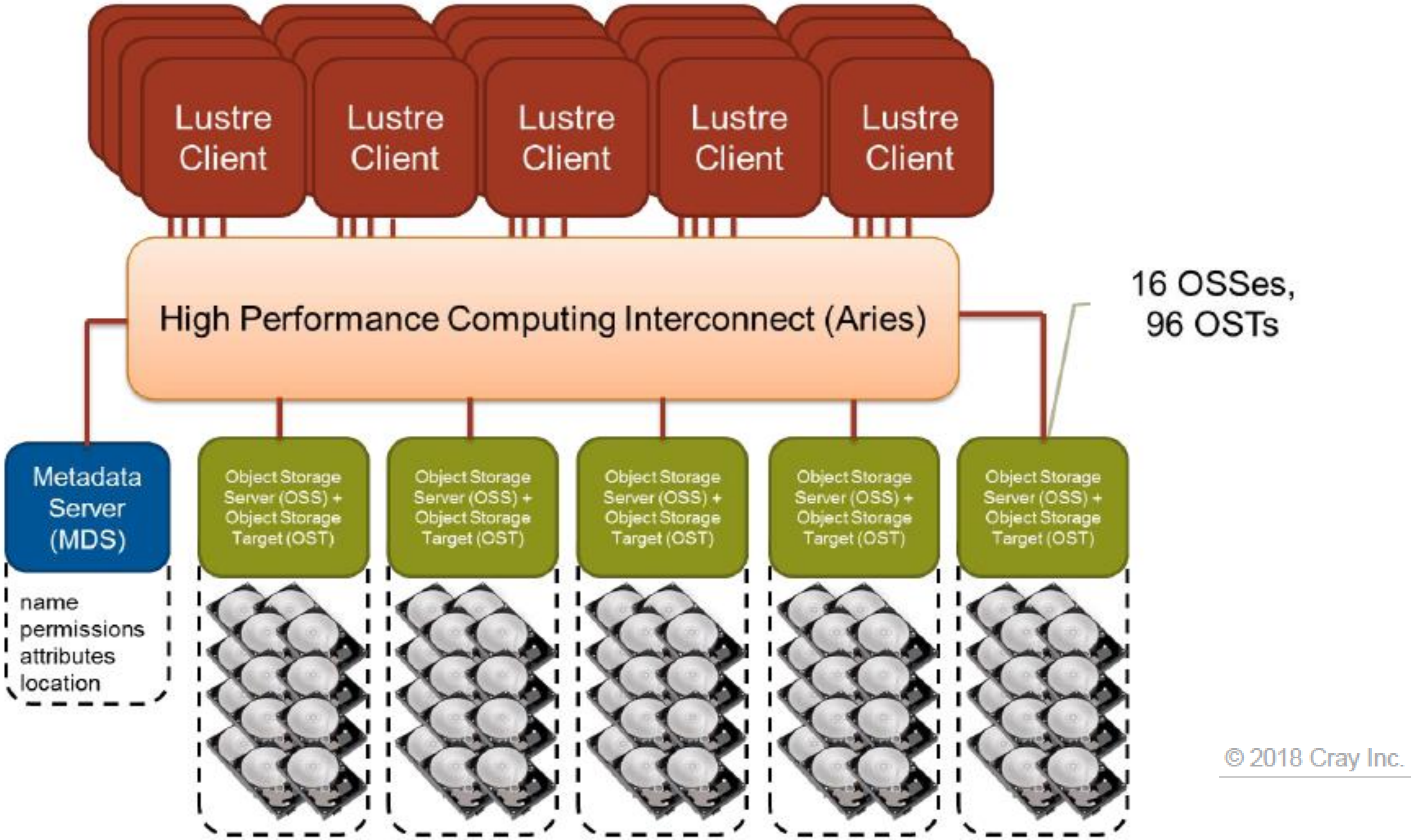
```
crayadm@login1:~> lfs df /mnt/lustre/
UUID                   1K-blocks       Used   Available Use% Mounted on
lustre-MDT0000_UUID    878145980   54298136   765294708   7% /mnt/lustre[MDT:0]
```

LUSTRE COMPONENTS …

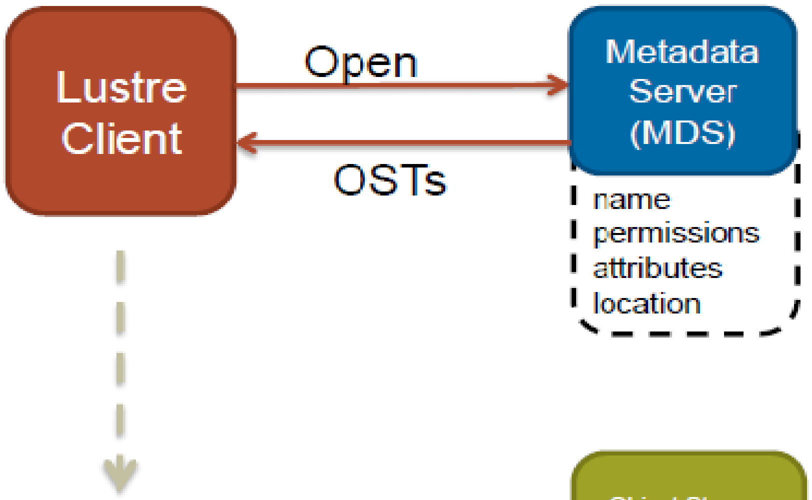
- **Object Storage Server (OSS)**
  - Communicates over a network
  - Provides file data services (objects)
  - Uses locally attached storage to store file data
    - Object Storage Metadata Target (OST)
    - An OSS can have one or more OSTs
- /mnt/lustre has 96 OSTs on 16 OSSes (6 OSTs per OSS)

```
crayadm@login1:~> lfs df /mnt/lustre/ | grep OST
lustre-OST0000_UUID  22935567680 12488481240  9299530652  57% /mnt/lustre[OST:0]
lustre-OST0001_UUID  22935567680 11053373400 10734551704  51% /mnt/lustre[OST:1]
```

LUSTRE ARCHITECTURE ON SAHASRAT AT SERC (CRAY XC-40)



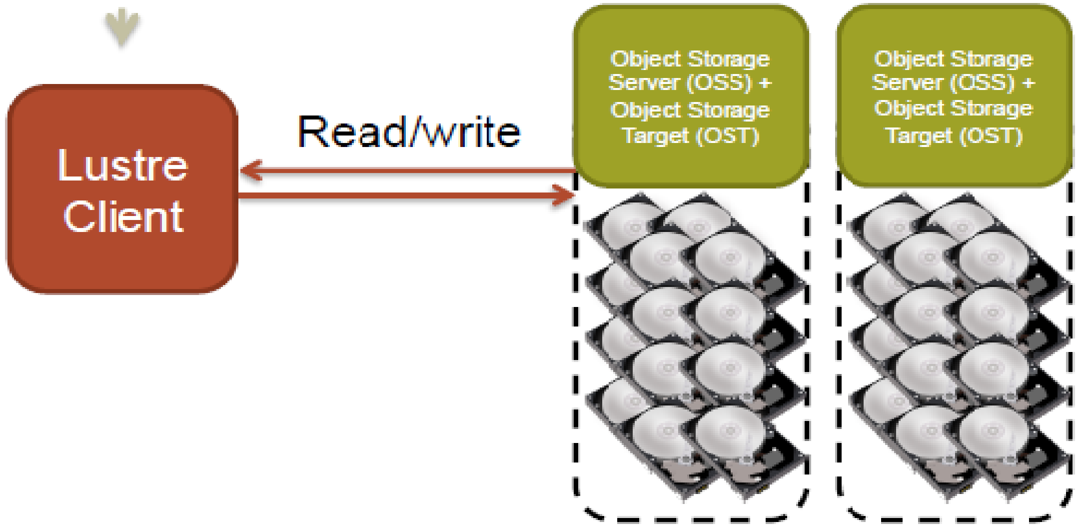
OPENING A FILE



The client sends a request to the MDS to opening/acquiring information about the file

The MDS then passes back a list of OSTs

- For an existing file, these contain the data stripes
- For a new files, these typically contain a randomly assigned list of OSTs where data is to be stored



Once a file has been opened no further communication is required between the client and the MDS

All transfer is directly between the assigned OSTs and the client

LUSTRE AND HIGH AVAILABILITY

- **Each Lustre file system comprises, at a minimum:**
  - 1 Management service (MGS), with corresponding Management Target (MGT) storage
  - 1 or more Metadata service (MDS) with Metadata Target (MDT) storage
  - 1 or more Object storage service (OSS), with Object Storage Target (OST) storage
- **For High Availability, the minimum working configuration is:**
- 2 Metadata servers, running MGS and MDS in failover configuration
  - MGS service on one node, MDS service on the other node
  - Shared storage for the MGT and MDT
- 2 Object storage servers, running multiple OSTs in failover configuration
  - Shared storage for the OSTs
  - All OSTs evenly balanced across the OSS servers

LUSTRE AND HIGH AVAILABILITY

- Every major enterprise operating system offers a high-availability cluster software framework
- Red Hat Enterprise Linux (RHEL) makes use of PCS (Pacemaker/Corosync Configuration System)
- SuSE Linux Enterprise Server (SLES) has CRMSH (Cluster Resource Management Shell)
- Both PCS and CRMSH are open-source applications

SUMMARY

- Large-scale data-intensive supercomputing relies on parallel file systems, such as Lustre, GPFS, PVFS etc. for high-performance I/O (Huaiming Song et al. 2011)
- I/O performance is a critical aspect of data-intensive scientific computing (Glenn K. Lockwood et al., 2018)
- Parallel I/O is one technique used to access data on disk simultaneously from different application processes to maximize bandwidth and speed things up (The HDF Group)
- Parallel I/O is a subset of parallel computing that performs multiple input/output operations simultaneously

ONLINE RESOURCES

- Introduction to Lustre: http://wiki.lustre.org/Introduction_to_Lustre
- Introduction to Lustre* Architecture: http://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf
- The NetCDF Tutorial: http://www.unidata.ucar.edu/software/netcdf/docs/netcdftutorial.pdf
- Introduction to HDF5: http://ww.hdfgroup.org/HDF5/doc/H5.intro.html
- The HDF group: https://www.hdfgroup.org/2015/04/parallel-io-why-how-and-where-to-hdf5/
- Parallel I/O Techniques and Performance Optimization: https://www.nics.tennessee.edu/sites/www.nics.tennessee.edu/files/pdf/Lonnie.pdf
- Parallel I/O in Practice: http://www.eecs.ucf.edu/~jwang/Teaching/EEL6760-f13/M02.tutorial.pdf
- Parallel file system: https://searchstorage.techtarget.com/definition/parallel-file-system
- Introduction to Parallel I/O: https://www.olcf.ornl.gov/wp-content/uploads/2011/10/Fall_IO.pdf

ONLINE RESOURCES …

- Parallel File Systems: http://www.cs.iit.edu/~iraicu/teaching/CS554-F13/lecture17-pfs-sam-lang.pdf
- Parallel I/O and MPI-IO: http://www.training.prace-ri.eu/uploads/tx_pracetmo/pio1.pdf
- Overview of Luster File System and I/O strategies: http://www.serc.iisc.ac.in/serc_web/wp-content/uploads/2018/01/SERC_IO_Workshop_Day1.pdf
- LUSTRE OVERVIEW: https://indico.fnal.gov/event/2538/session/27/contribution/17/material/slides/1.pdf
- Advanced MPI Techniques: http://morrisriedel.de/wp-content/uploads/2018/03/HPC-Lecture-4-HPC-Advanced-MPI-Techniques-Public.pdf
- Architecture of a Next-Generation Parallel File System: https://events.static.linuxfound.org/images/stories/pdf/lfcs2012_wilson.pdf
- High Level Introduction to HDF5: https://support.hdfgroup.org/HDF5/Tutor/HDF5Intro.pdf

Thank you