PARALLEL FILE SYSTEMS

Filbert Minj

Storage Team

SERC, Indian Institute of Science

filbert@iisc.ac.in

OUTLINE OF THE CONTENTS

- File Systems
- Parallel File Systems
- Parallel vs. Distributed
- Storage Device
- What is Parallel I/O?
- Parallel I/O Tools
- High Level Libraries
- I/O Middleware
- MPI-IO
- Parallel File Systems and Performance
- Parallel File System Architectures
- Lustre Overview
- Summary
- References

SERC STORAGE SPACE

- Cray XC-40 (SahasraT) SUPERCOMPUTER
  - 2 PB of storage space
  - Used for scratch space
- NFS File server
  - 100 GB space
  - Provides user’s home area

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, ext3 on laptop
- Networked file systems provide access to one or more clients who might not have direct access to the disk
  - e.g. NFS, AFS, etc.

WHAT IS PARALLEL FILE SYSTEMS

- A parallel file system is a software component designed to store data across multiple networked servers and to facilitate high-performance access through simultaneous, coordinated input/output operations (IOPS) between clients and storage nodes
- Breaks up a data set and distributes, or stripes, the blocks to multiple storage drives, which can be located in local and/or remote servers
- Uses a global namespace to facilitate data access often use a metadata server to store information about the data, such as the file name, location and owner
- Reads and writes data to distributed storage devices using multiple Input/Output(I/O) paths concurrently and provide a significant performance benefit
- Capacity and bandwidth can be scaled to accommodate enormous quantities of data

WHY PARALLEL FILE SYSTEMS?

- HPC and Big Data applications increasingly rely on I/O subsystems
  - Large input datasets, checkpointing, visualization
- Programmers need interfaces that match their problem
  - Multidimensional arrays, typed data, portable formats
- Two issues to be resolved by I/O system
  - Performance requirements (concurrent access to HW)
  - Gap between app. abstractions and HW abstractions
- Software is required to address both of these problems

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

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)

PARALLEL VS. DISTRIBUTED

- How are Parallel File Systems different from Distributed File Systems?
- Data distribution
  - Distributed file systems often store objects (files) on a single storage node
  - Parallel file systems distribute data of a single object across muliple storage nodes
- Symmetry
  - Distributed file systems often run on architectures where the storage is co-located with the application (not always, e.g. GoogleFS, Ceph)
  - Parallel file systems are often run on architectures where storage is physically separate from the compute system (not always true here either)
- Fault-tolerance
  - Distributed file systems take on fault-tolerance responsibilities
  - Parallel file systems run on enterprise shared storage
- Workloads
  - Distributed file systems are geared for loosely coupled, distributed applications
  - Parallel file systems target HPC applications, which tend to perform highly coordinated I/O accesses, and have massive bandwidth requirements

WHAT MEANS I/O?

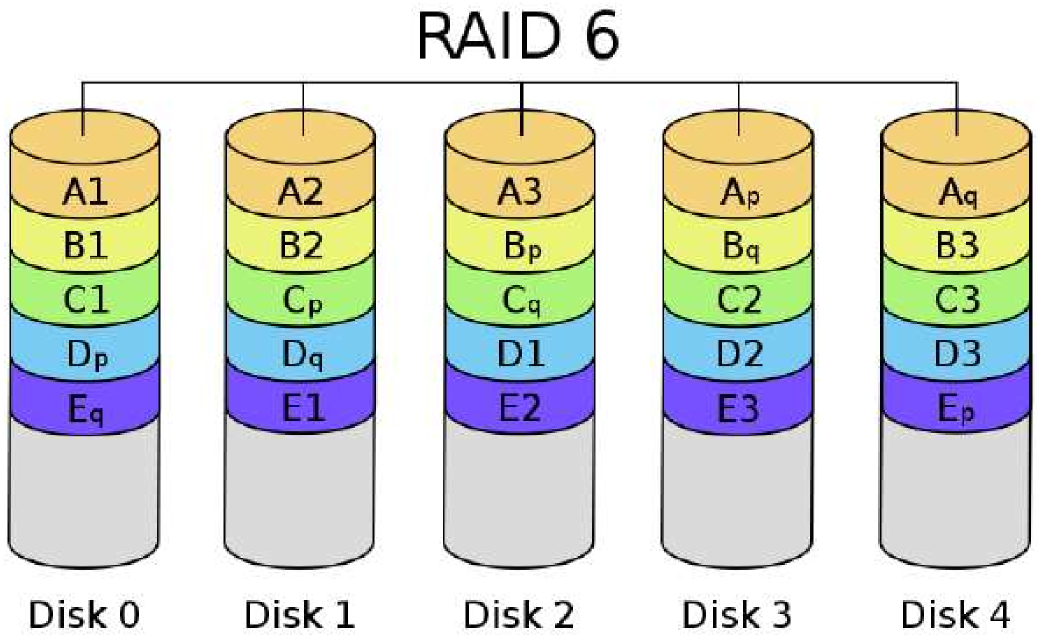
- Input/Output(I/O) stands for data transfer/migration from memory to disk (or vice versa)
- Important (time-sensitive) factors within HPC environments
  - Characteristics of the computational system (e.g. dedicated I/O nodes)
  - Characteristics of the underlying filesystem (e.g. parallel file systems,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

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 is typically used for distributed, parallel storage

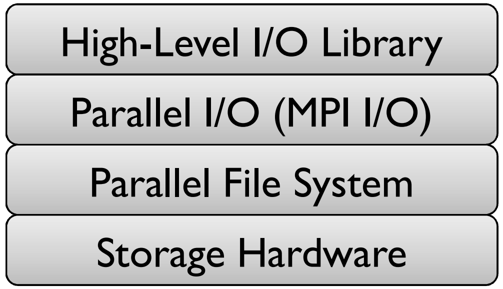
STORAGE DEVICE

- 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
  - ➢ 8+2 typical 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



CLASSES OF PARALLEL FILE SYSTEMS: BLOCKS VS. OBJECTS

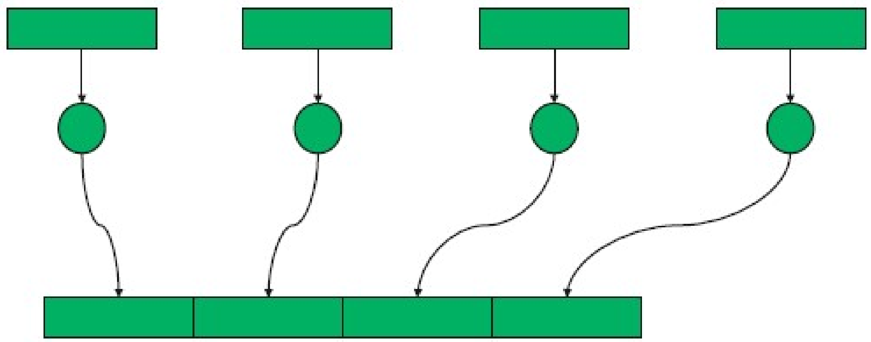
- Block-Based Parallel File Systems (AKA "Shared-disk")
  - Blocks are fixed-width
  - File growth requires more blocks
  - Blocks distributed over storage nodes
  - Suffer from block allocation issues, lock managers
  - Example: GPFS
- Object-based Parallel File Systems
  - Variable-length regions of the file
  - A file has a constant number of objects
  - Objects are given global identifiers (object-ids, handles, etc.)
  - File growth increases the size of object(s)
  - Objects are easier to manage and distribute
  - Space allocation is managed locally on a per-object basis
  - Examples: Lustre, PVFS

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

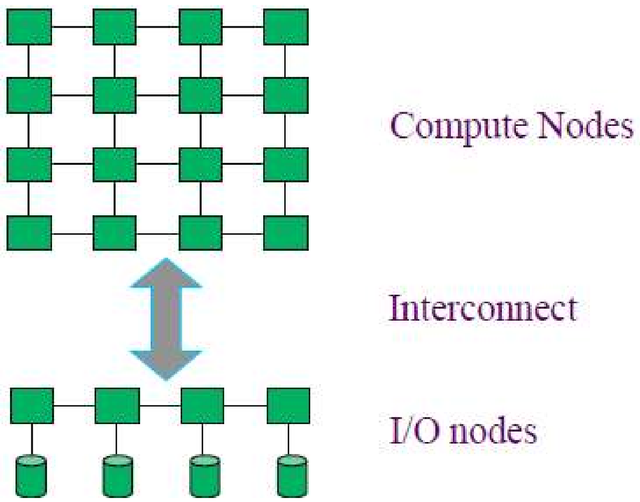
WHAT IS PARALLEL I/O?

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



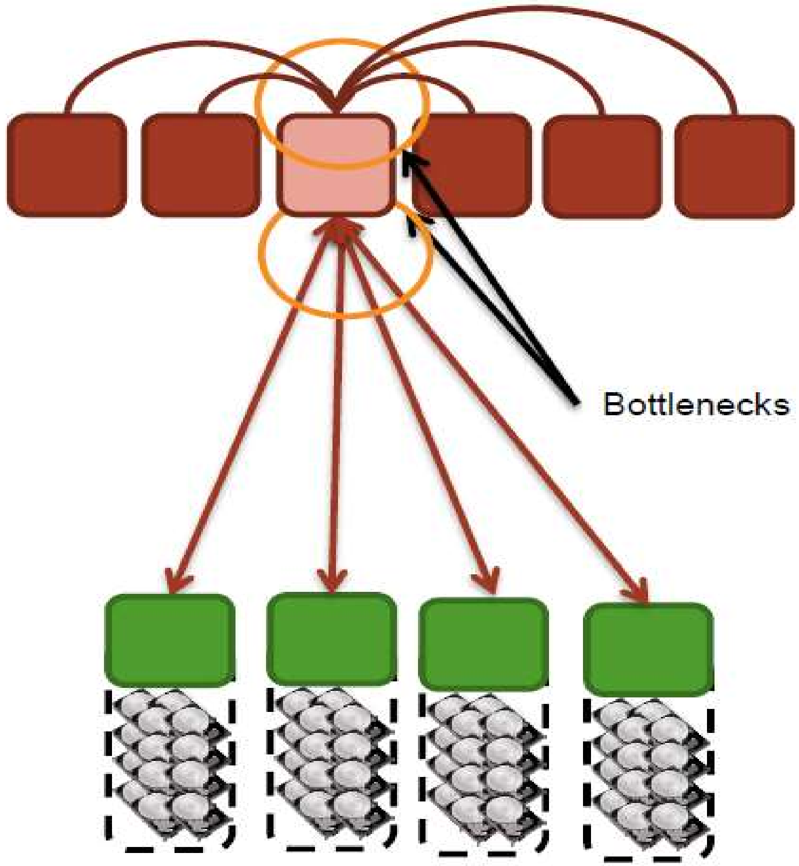
WHAT IS PARALLEL I/O? …

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



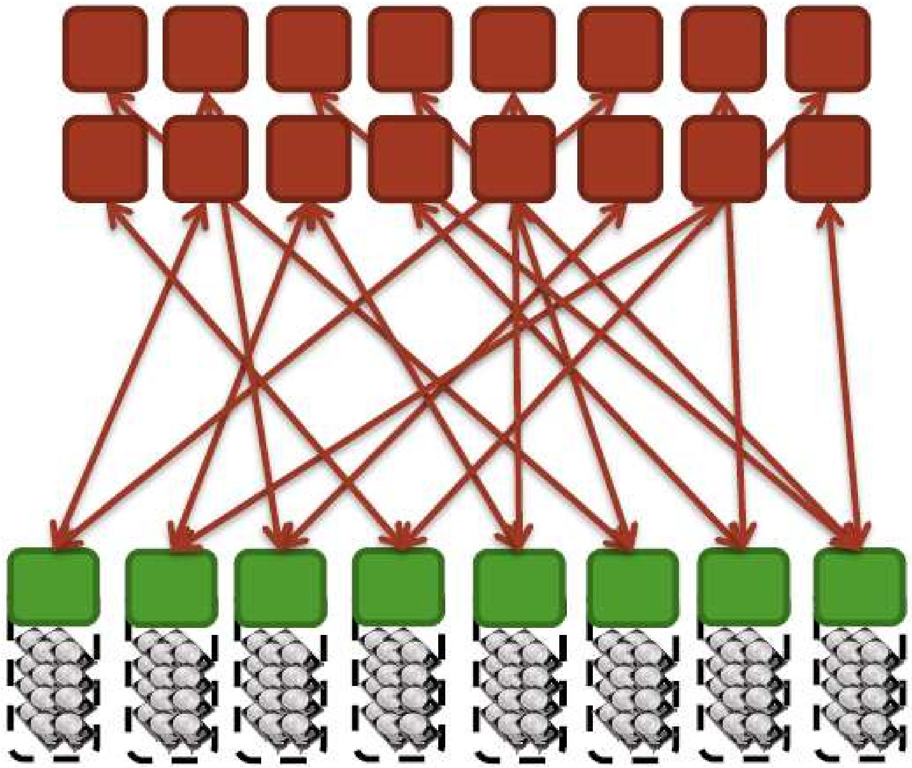
SERIAL I/O: SPOKESPERSON

- One process performs I/O.
  - Data aggregation or duplication
  - Limited by single I/O process
- Simple solution, easy to manage, but
  - Pattern does not scale
    - Time increases linearly with amount of data
    - Time increases with number of processes



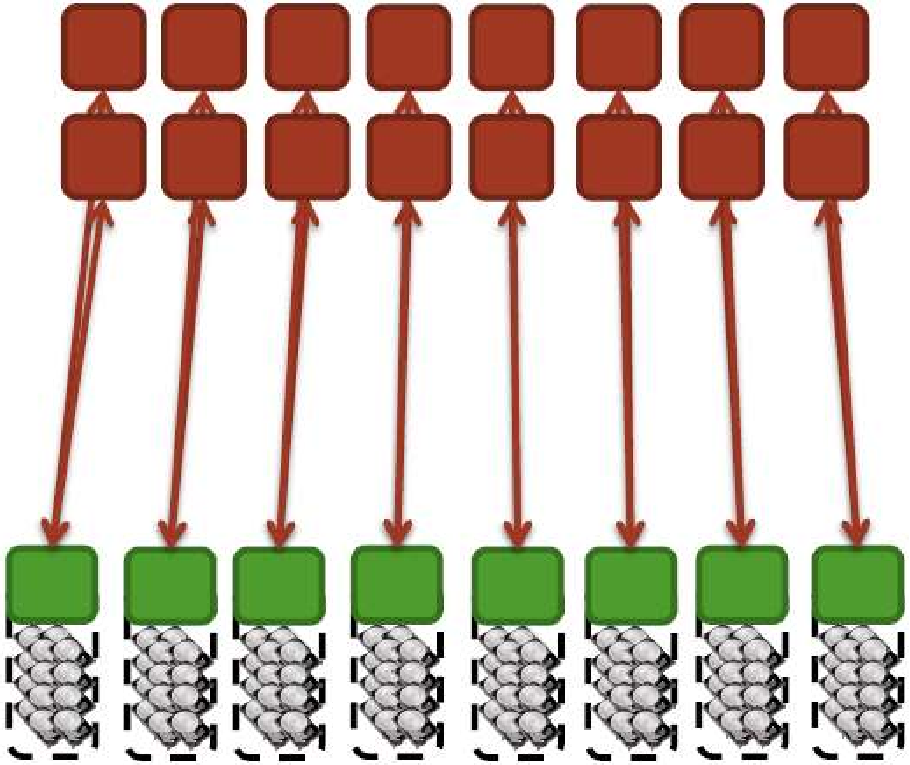
PARALLEL I/O: FILE-PER-PROCESS

- All processes perform I/O to individual files
  - ➢ Limited by file system
- Pattern does 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



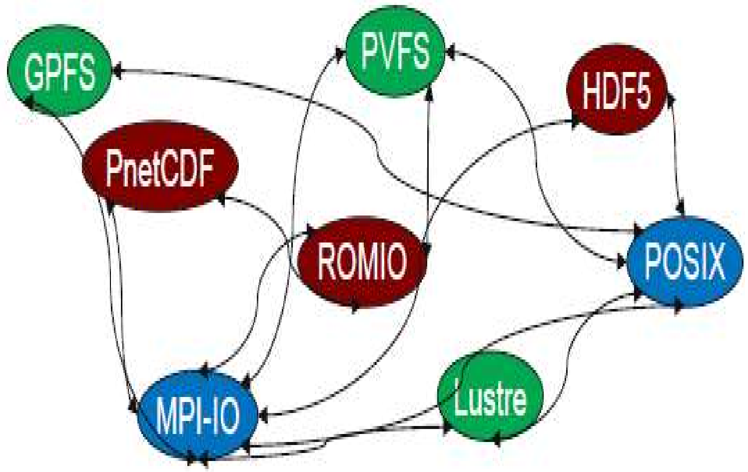
PARALLEL I/O: SHARED FILE

- Shared File
  - Each process performs I/O to a single file which is shared
  - Performance
    - Data layout within the shared file is very important
    - At large process counts contention can build for file system resources
  - Not all programming languages support it

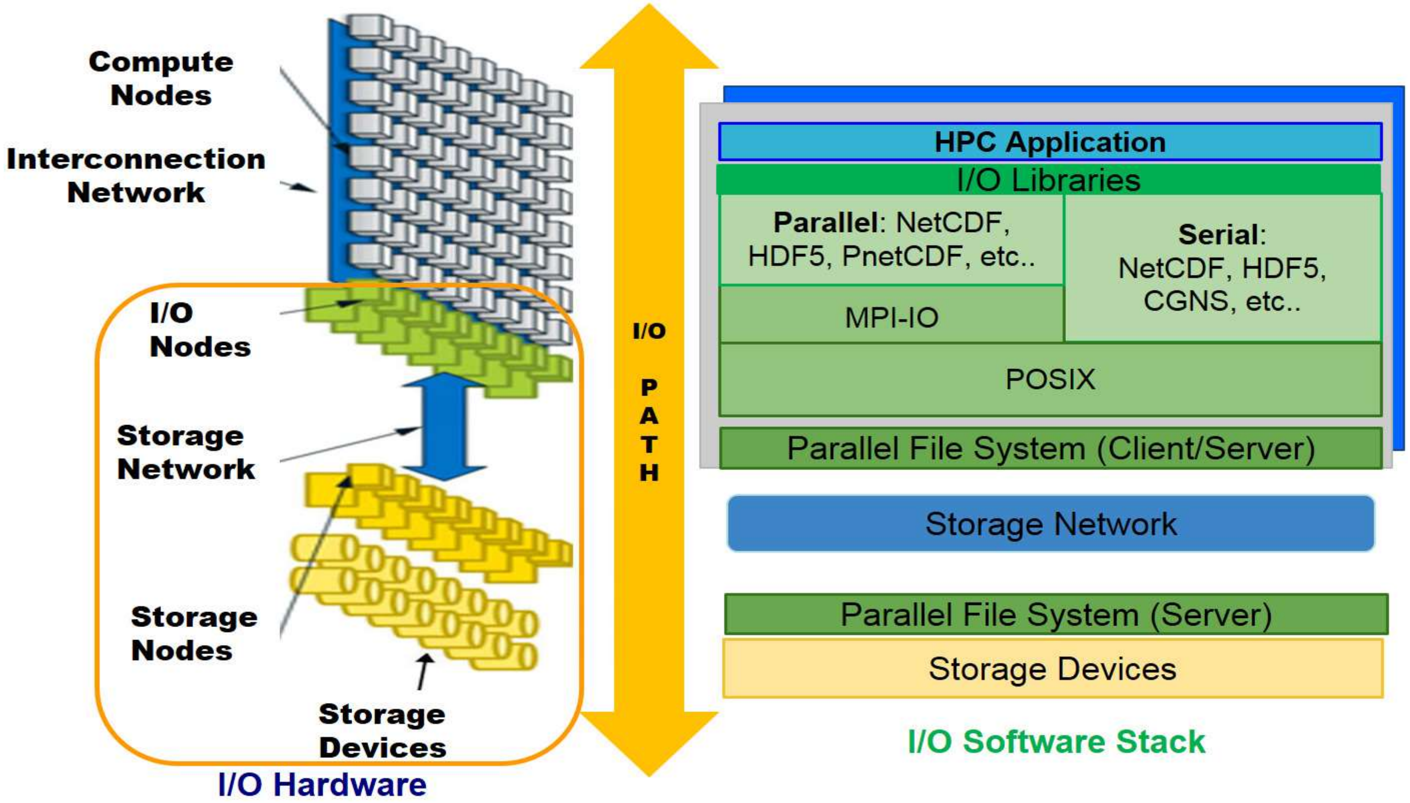


PARALLEL I/O TOOLS

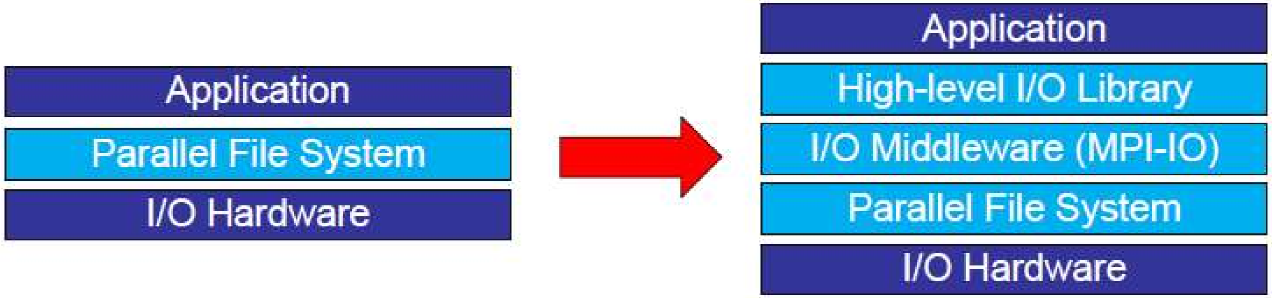
- System software and libraries have grown up to address I/O issues
  - Parallel file systems
  - MPI-IO (Message Passing Interface)
  - High level libraries
- Relationships between these are not always clear
- Choosing between tools can be difficult



PARPARALLEL I/O TOOLS



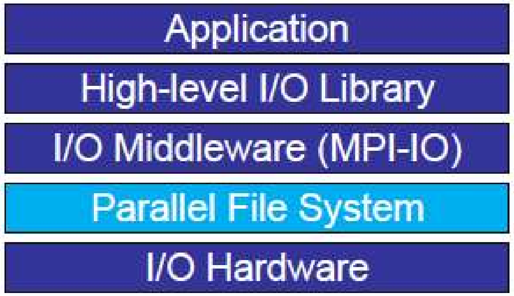
PARALLEL I/O TOOLS FOR COMPUTATIONAL SCIENCE



- 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
  - 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
  - Scalable

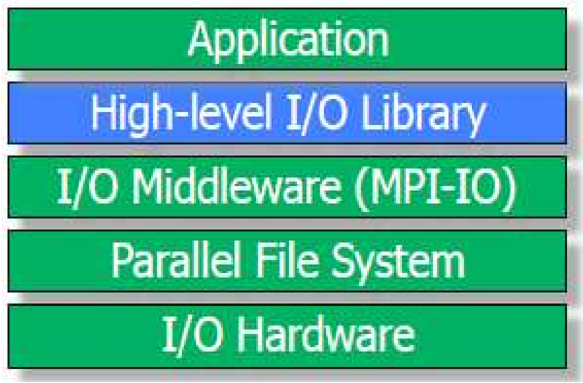


I/O MIDDLEWARE

- Facilitate concurrent access by groups of processes
- 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



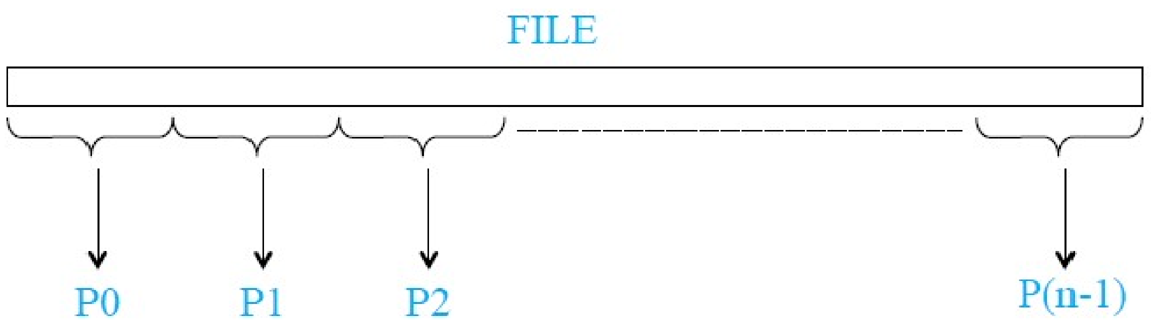
HIGH LEVEL I/O LIBRARIES (HDF5)

- HDF5 = Hierarchical Data Format, v5
- Open file format
  - ➢ Designed for high volume or complex data
- Open source software
  - ➢ Works with data in the format
- An extensible data model
  - ➢ Structures for data organization and specification

MPI-IO

- I/O interface specification for use in MPI apps
- Data Model:
  - Stream of bytes in a file
  - Portable data format (external32)
    - Not self-describing - just a well-defined encoding of types
- Features:
  - Collective I/O
  - Noncontiguous I/O with MPI datatypes and file views
  - Nonblocking I/O
  - Fortran bindings (and additional languages)
- Implementations available on most platforms

USING MPI FOR SIMPLE I/O



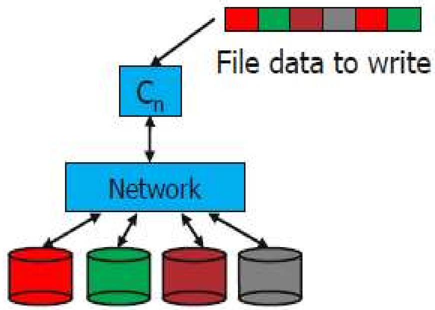
Each process needs to read a chunk of data from a common file

WHY MPI IS A GOOD SETTING FOR PARALLEL I/O

- Writing is like sending and reading is like receiving
- Any parallel I/O system will need:
  - collective operations
  - user-defined datatypes to describe both memory and file layout
  - communicators to separate application-level message passing from I/O-related message passing
  - non-blocking operations
  - i.e. lots of MPI-like machinery

PARALLEL FILE SYSTEMS AND PERFORMANCE

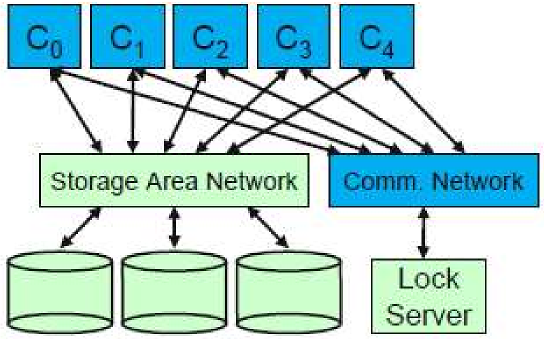
- Striping is the basic mechanism used in parallel file system to improve performance
  - ➢ Striping refers to a technique where one file is split into fixed-sized blocks that are written to separate disks in order to facilitate parallel access
- Primarily striping allows multiple servers, disks, network links to be leveraged during concurrent I/O operations
  - ➢ Eliminates bottlenecks
  - ➢ Can also improve serial performance over a single, local disk
- Coordinating access can re-introduce bottlenecks
  - ➢ But is necessary for coherence



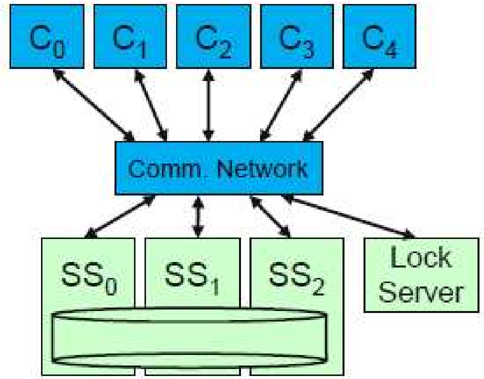
PARALLEL FILE SYSTEM ARCHITECTURES

- 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

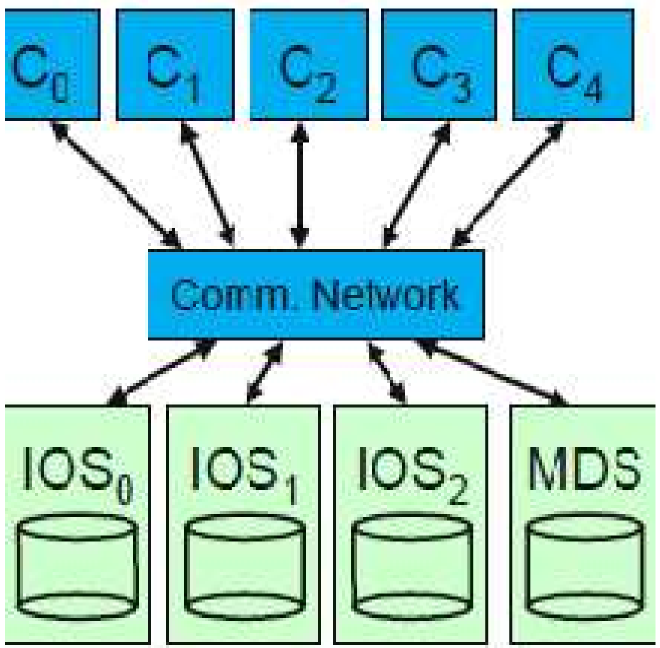


Pooled storage using existing interconnect

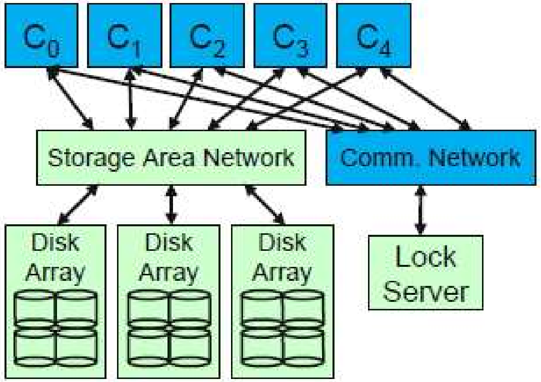
- Clients share access to disk blocks on real or virtual disks
  - Directly via Fibre-Channel SAN, iSCSI, AT over Ethernet
  - Indirectly via storage servers
    - e.g. Virtual Shared Disk, Network Shared Disk
    - May expose devices directly, or pool them into a larger whole
- Lock server coordinates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

OBJECT SERVER ARCHITECTURES

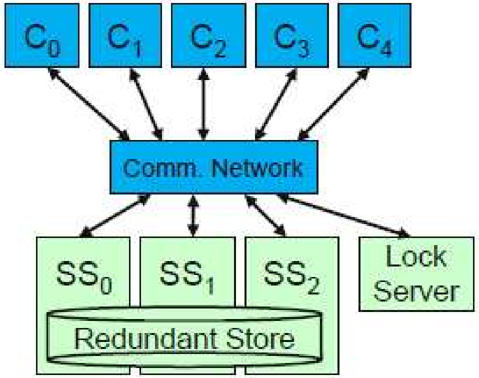
- Clients share access to files or objects
- Servers are "smart"
  - Understand something about the structure of data on storage
  - I/O servers (IOS) manage local storage allocation
    - Map client accesses into local storage operations
- Metadata server (MDS) stores directory and file metadata
  - 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 SHARED STORAGE



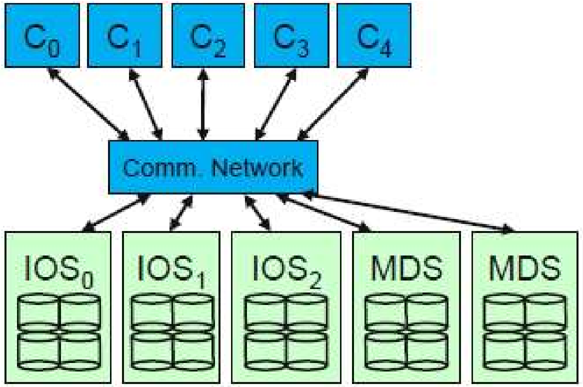
Redundancy with directly attached storage



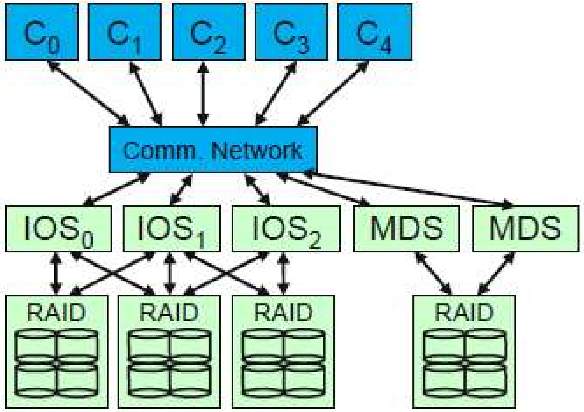
Redundancy with virtual shared storage

- For directly attached storage
  - Single disk array can provide hardware redundancy
  - Clients can stripe data across multiple disk arrays
- For virtual shared storage
  - Storage servers replicate blocks, store redundant data across physical resources
  - SAN may be used behind storage servers for connectivity

REDUNDANCY WITH OBJECT SERVER



Redundancy with replicated local storage

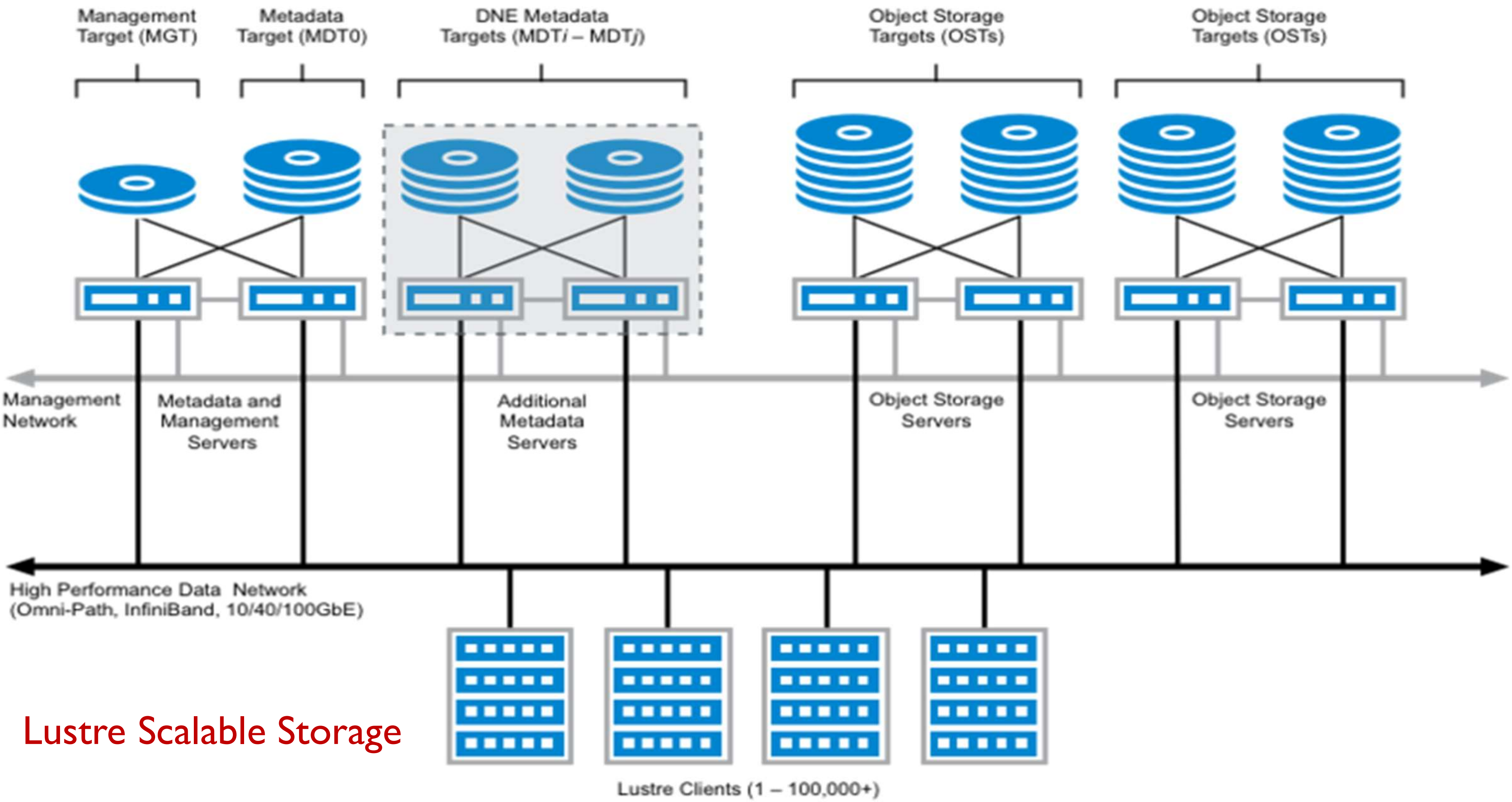


Redundant storage connectivity for failover

- 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 OVERVIEW

- Open source object-based parallel file system
- Global single-namespace
- POSIX-compliant (Portable Operating System Interface)
- Distributed parallel file system designed for scalability, high-performance, and high-availability
- Lustre runs on Linux-based operating systems and employs a client-server network architecture



Lustre Scalable Storage

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 MDT(Metadata Target) to store metadata information
    - An MDS can have one or more MDTs
  - /mnt/lustre has one MDS and one MDT

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)

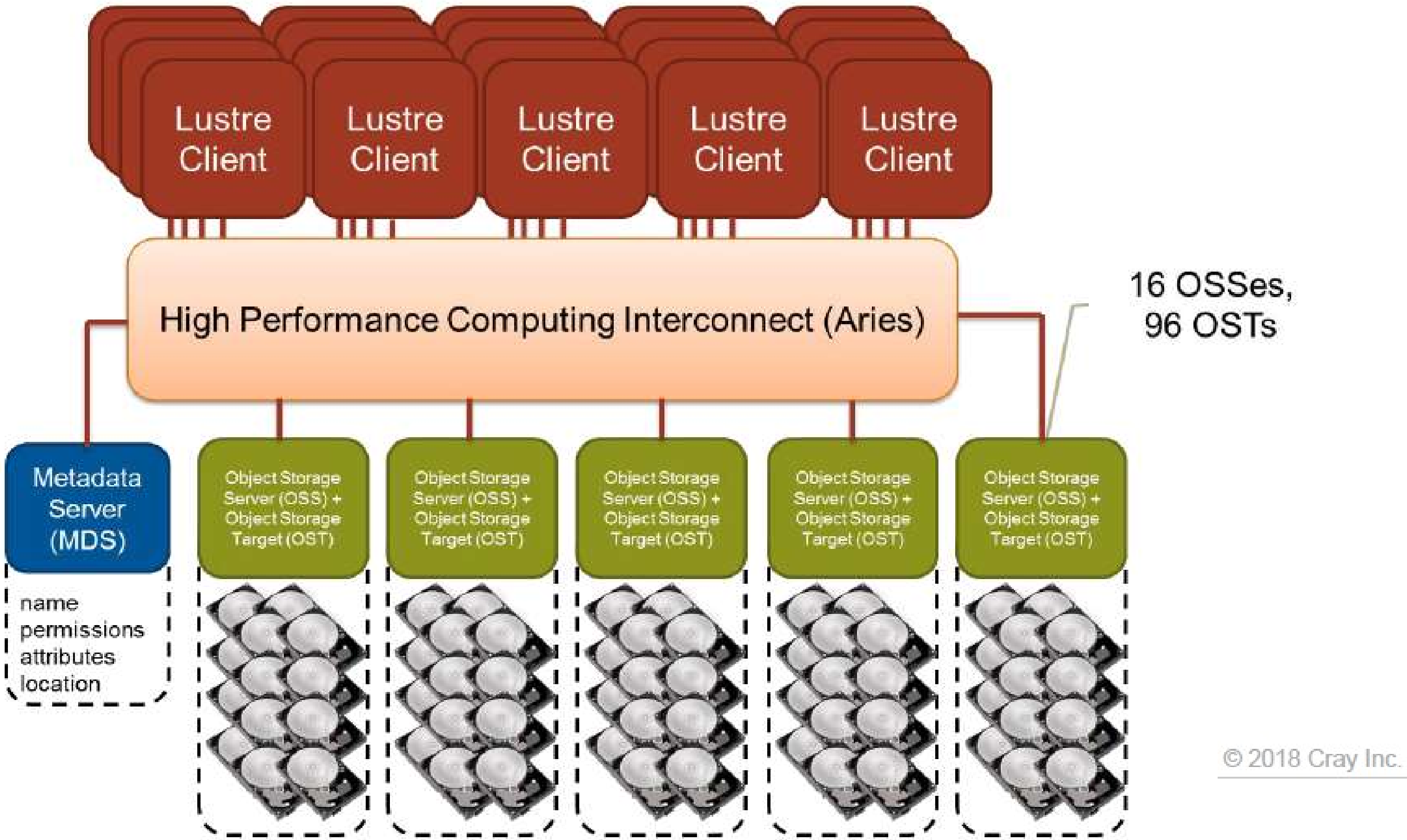
LUSTRE COMPONENTS …

- **Clients**
  - Lustre clients mount each Lustre file system instance using the Lustre Network protocol (LNet)
  - Presents a POSIX-compliant file system to the OS
  - Provides concurrent and coherent read and write access
  - Accesses MDS and OSS resources in parallel
  - Provides client caching capabilities

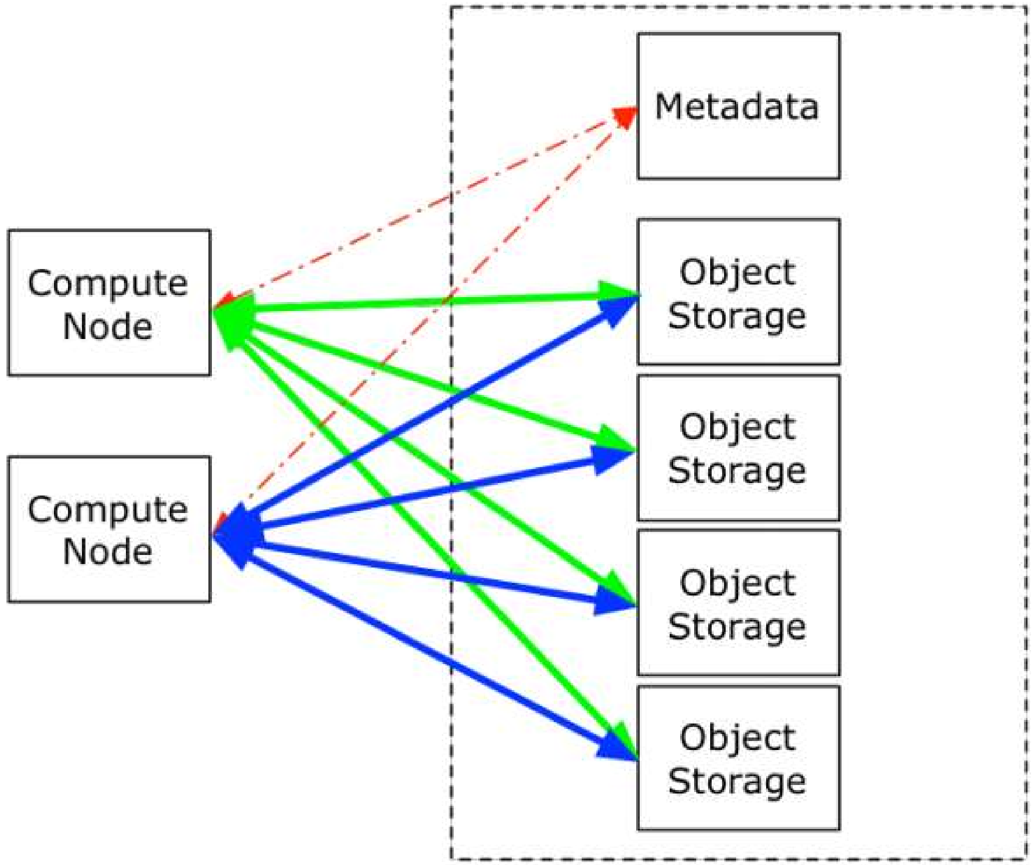
COMPONENTS INTO A WHOLE FILE SYSTEM

- Clients access metadata and object data by making requests to MDSes and OSSes
  - Clients do not directly modify data or metadata
- A distributed lock manager is used to provide coherency
  - Each OST manages locks for objects it contains
- A single file can be stored across many different OSTs
- How a file is distributed between OSTs is done by default settings or end-user requested behavior
  - Stripe Count: The number of OSTs the file should store stripes on
  - Stripe Size: The size of data that should be stored on a single OST before using the next OST
- Any client can place files on any OST

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



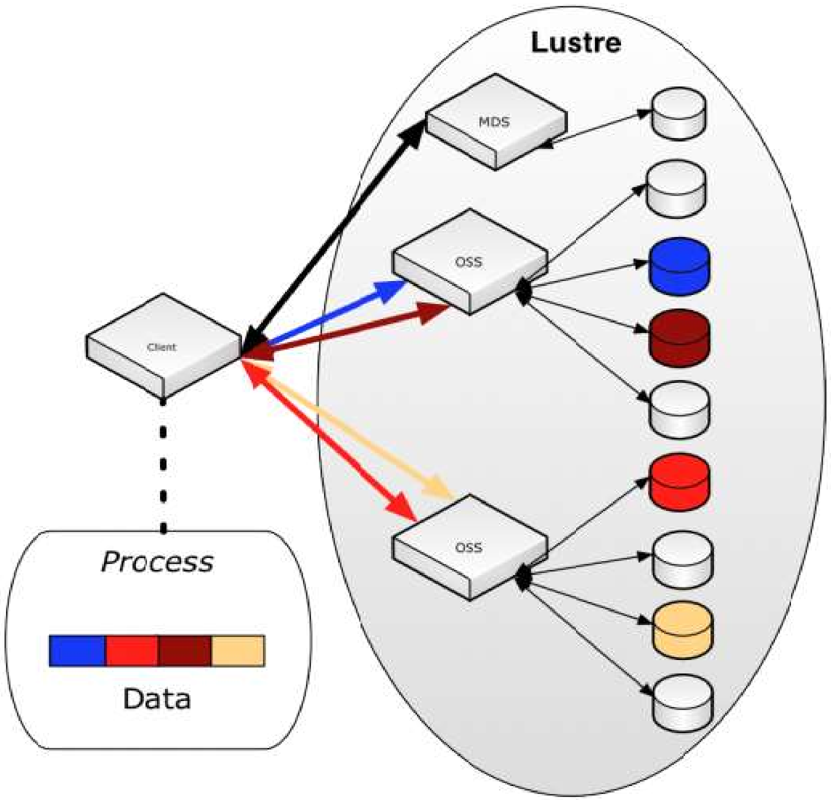
LUSTRE PARALLEL I/O



Stephen Simms, Indiana University

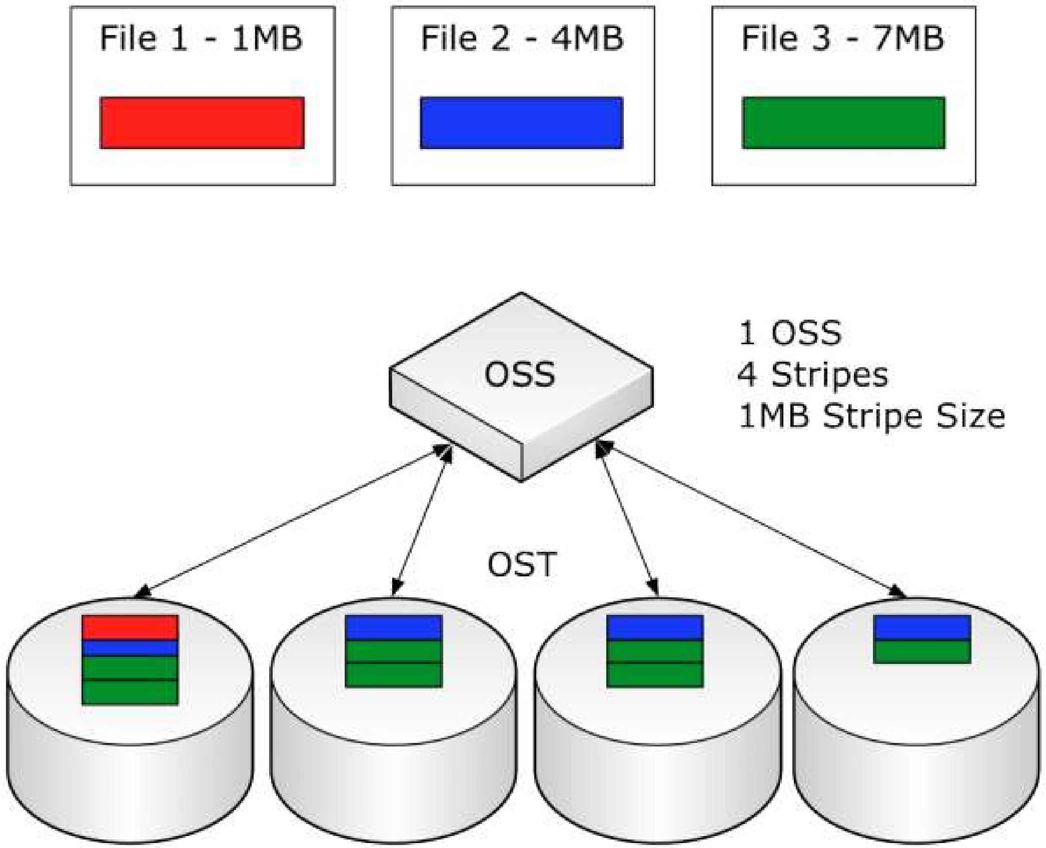
STRIPING DATA

- Lustre allows you to control how data is written, if you want
  - Stripe data across multiple OSTs
    - can stripe files OR directories
  - Can increase I/O performance with reading and writing
  - With DNE2 (Distributed Namespace Environment) metadata can be Striped across multiple MDTs
- Striping analogous to RAID 0
- Default striping set by sysadmin



Stephen Simms, Indiana University

STRIPING EXAMPLE



Stephen Simms, Indiana University

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
- HAWK (HIGH AVAILABILITY WEB KONSOLE) Web interface to CRMSH and PCS has its own web-based UI

LUSTRE AND HIGH AVAILABILITY

- MGS and MDS usually paired into a high availability server configuration
- Each Lustre file system comprises, at a minimum:
  - 1 x Management service (MGS, with MGT storage)
  - 1 x Metadata service (MDS, with MDT storage)
  - 1+ Object storage service (OSS, with 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 1, 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

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