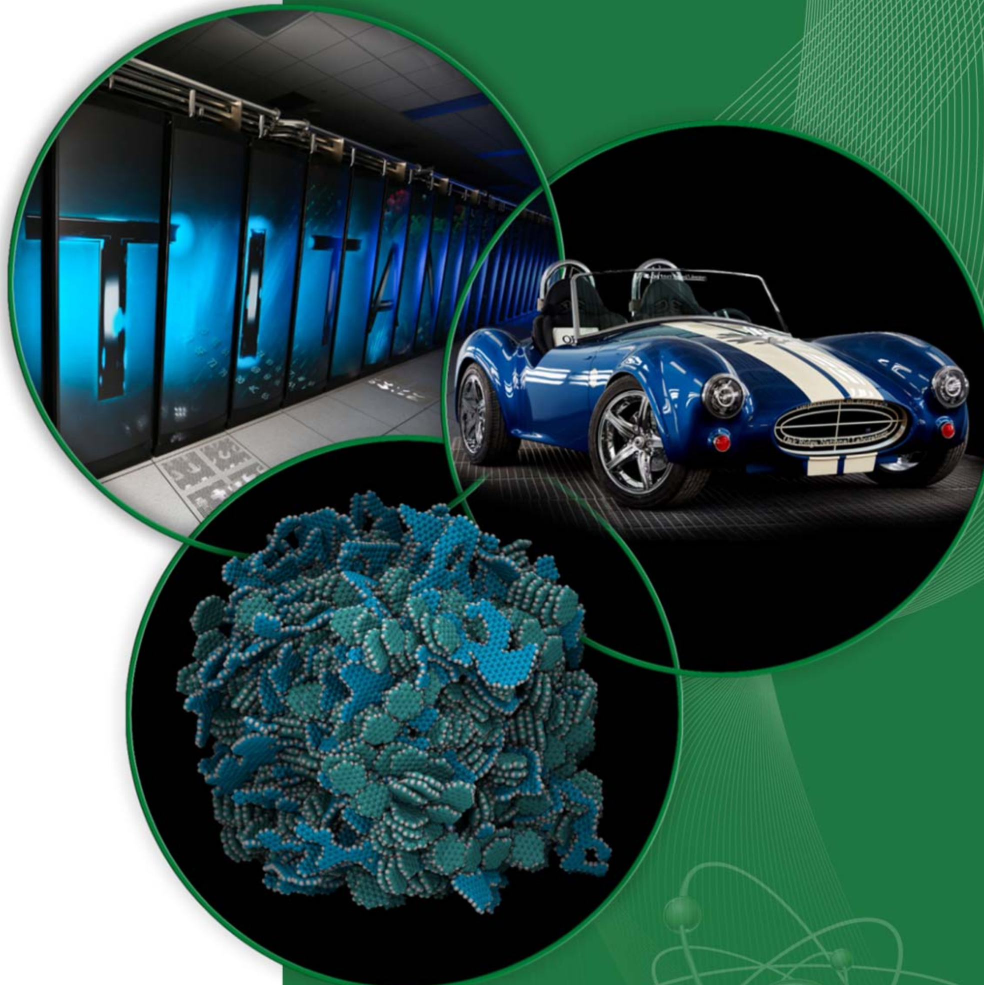
Introduction to HPC Parallel I/O



Feiyi Wang (Ph.D.) and Sarp Oral (Ph.D.)

Technology Integration Group

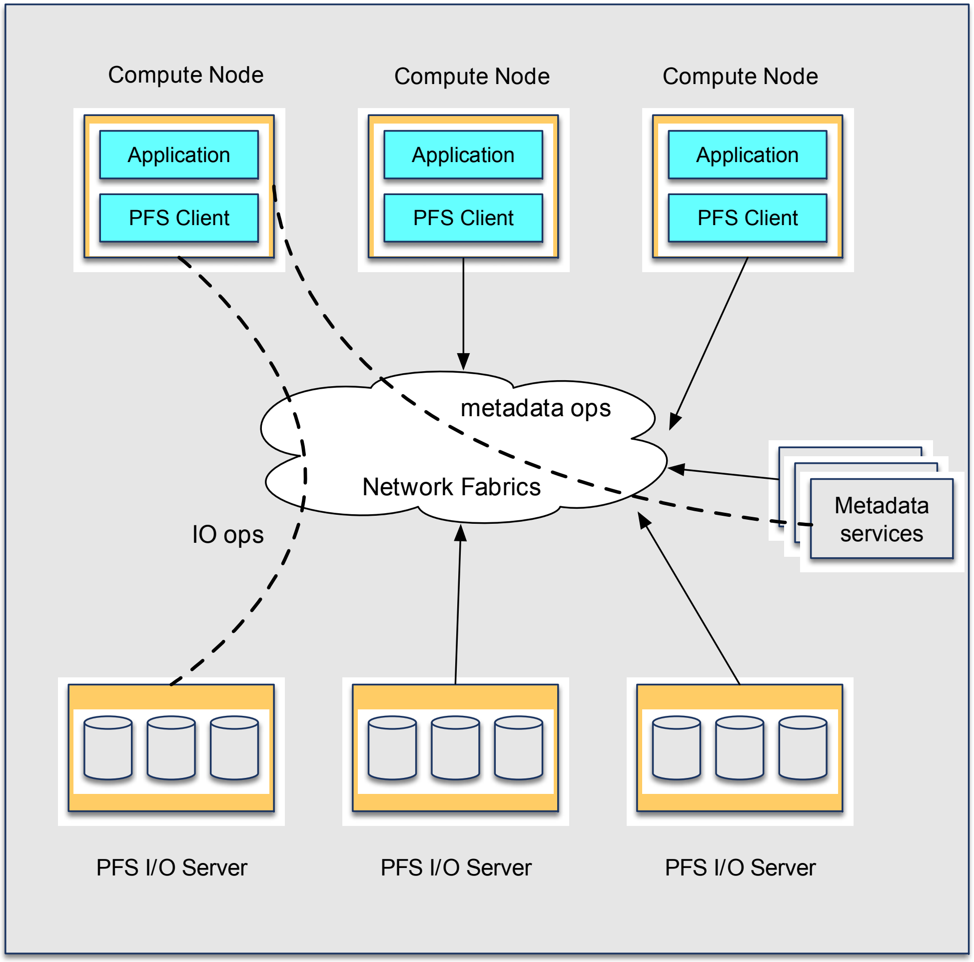
Oak Ridge Leadership Computing

Outline

- Parallel I/O in HPC
  - General concepts
  - I/O environment
  - Programming perspective
  - System perspective
  - Performance perspective
- Establish end-to-end system perspective and set the right expectation

HPC parallel I/O environment

- HPC I/O System is more than just (PFS) parallel file system, but PFS is the first user interaction point.



At very high-level, PFS is logically composed of three components:

- PFS clients
- metadata services
- I/O servers

PFS is also the first point to reveal any system problem

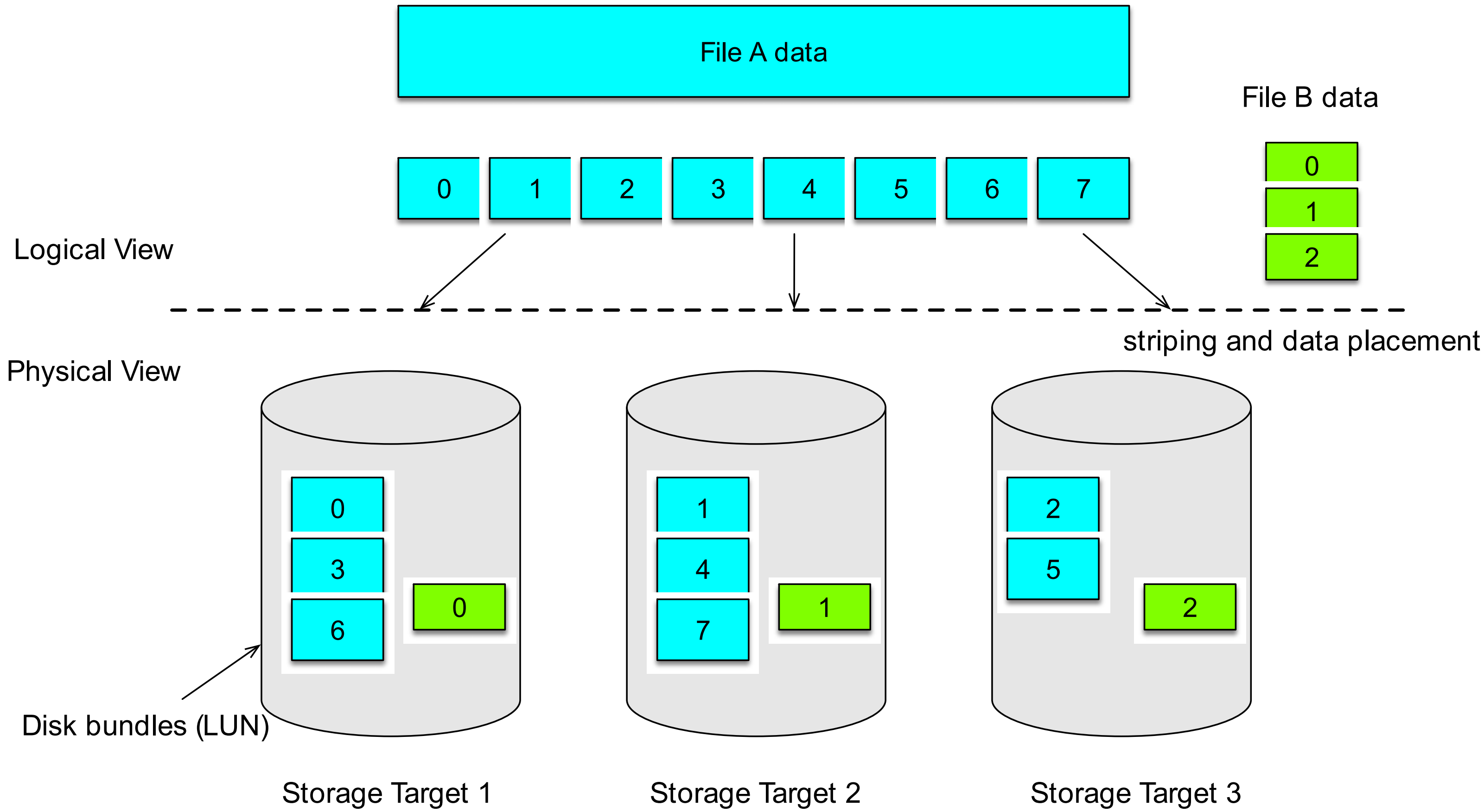
- Network
- Memory
- Storage

Oak Ridge National Laboratory

Parallel file system

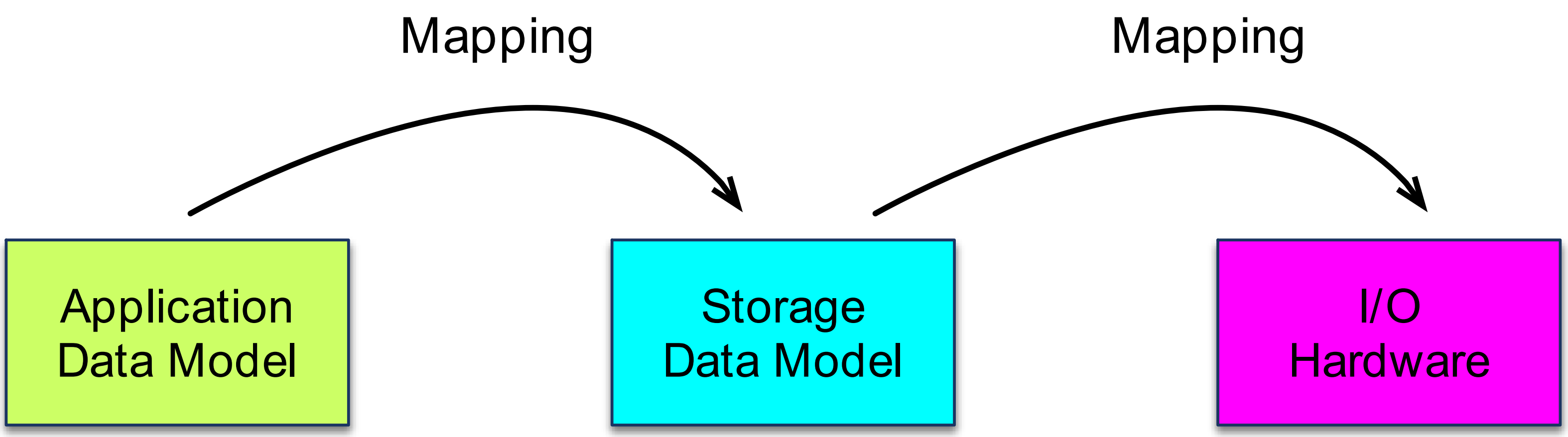
- Global namespace
- Designed for ***parallelism and scalability***
  - Concurrent access from tens of thousands clients
  - Efficient N-to-1 and N-to-N access pattern
- Designed for ***high-performance***
  - High-performance interconnect
  - High-performance protocol stack
- POSIX-compliant
- PFS is different from desktop/laptop file system
  - A contested and shared resource
  - Vastly more complex
  - Network attached.
- Common PFS flavors
  - Lustre, GPFS, Ceph, PanasasFS, PVFS, BeeGFS

I/O parallelization



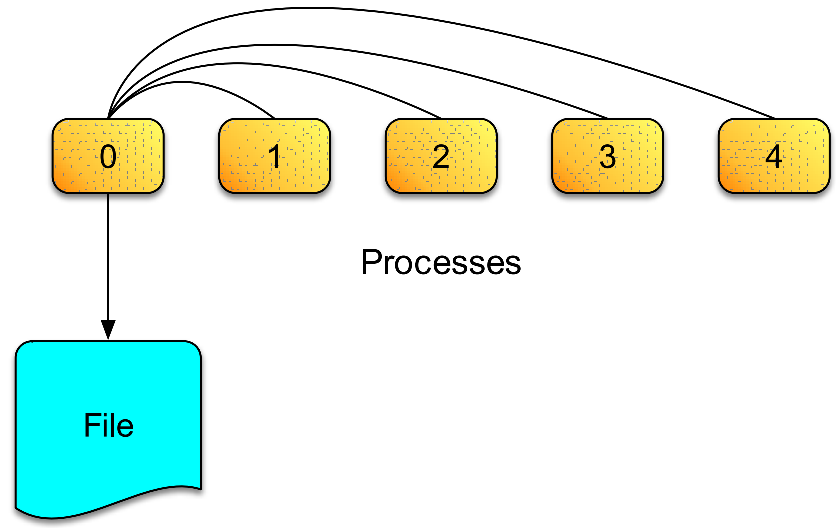
Programming perspective of parallel I/O

- Persist some amount of data as quickly as possible, as reliably as possible
- In HPC, we have a wide array of choices with language bindings for C, C++ and Fortran
  - POSIX I/O
  - MPI/IO
  - Parallel netCDF or HDF5
  - Middleware such as ADIOS



Programming perspective - serial I/O

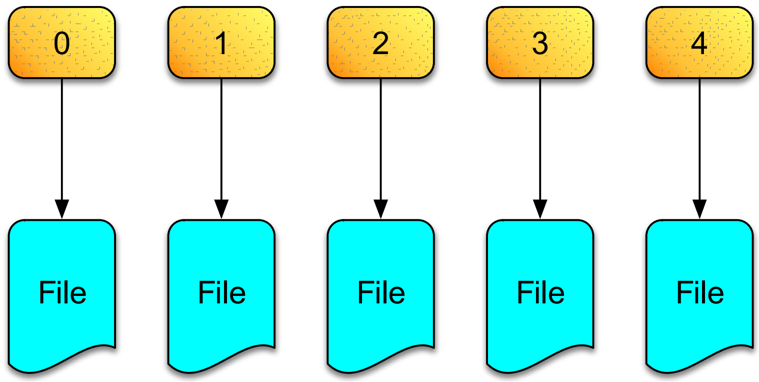
- Processes send data to the master
- Master writes the data to a file
- Read is in the reverse order



- Advantages
  - Simple
  - Good performance for very small I/O sizes
- Disadvantages
  - Not efficient
    - Two-stage process
  - Not scalable
    - Slow for any large number of processes or data sizes
    - May not be possible if memory constrained

Programming perspective - parallel I/O file per process

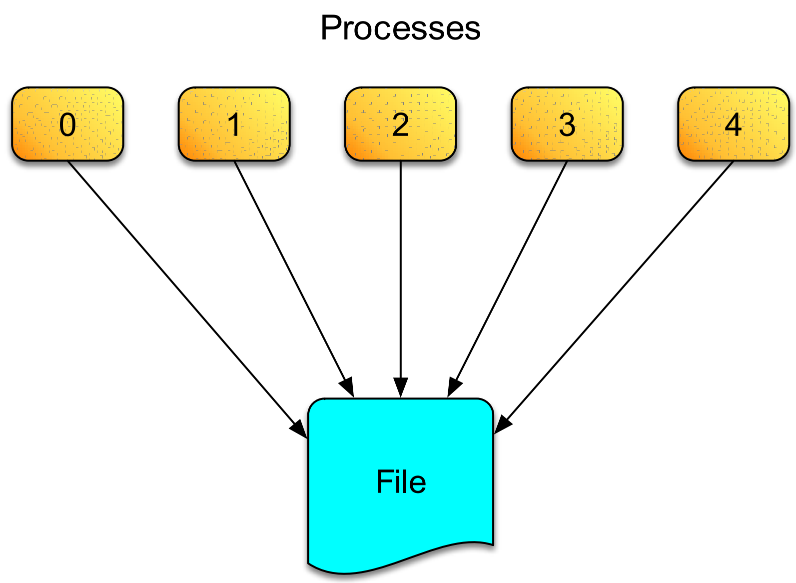
- Each process writes its own data to a separate file
- N to N
- Advantages
  - Simple to program
  - Can be fast



- Disadvantages
  - Impact on PFS metadata server – performance weak spot of single shared directory
  - Can quickly accumulate many files, hard to manage
  - Difficult for archival systems, e.g. HPSS, to handle many small files

Programming perspective - parallel I/O single shared file

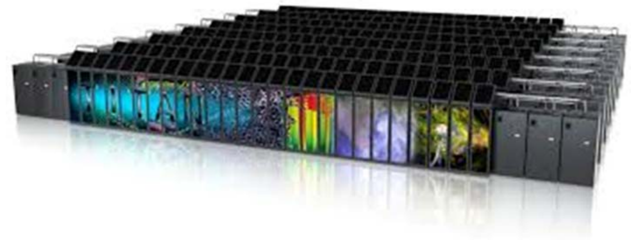
- Each process writes its own data to the same file using such as MPI-IO mapping
- N to 1
- Advantages
  - Single file
  - Manageable data

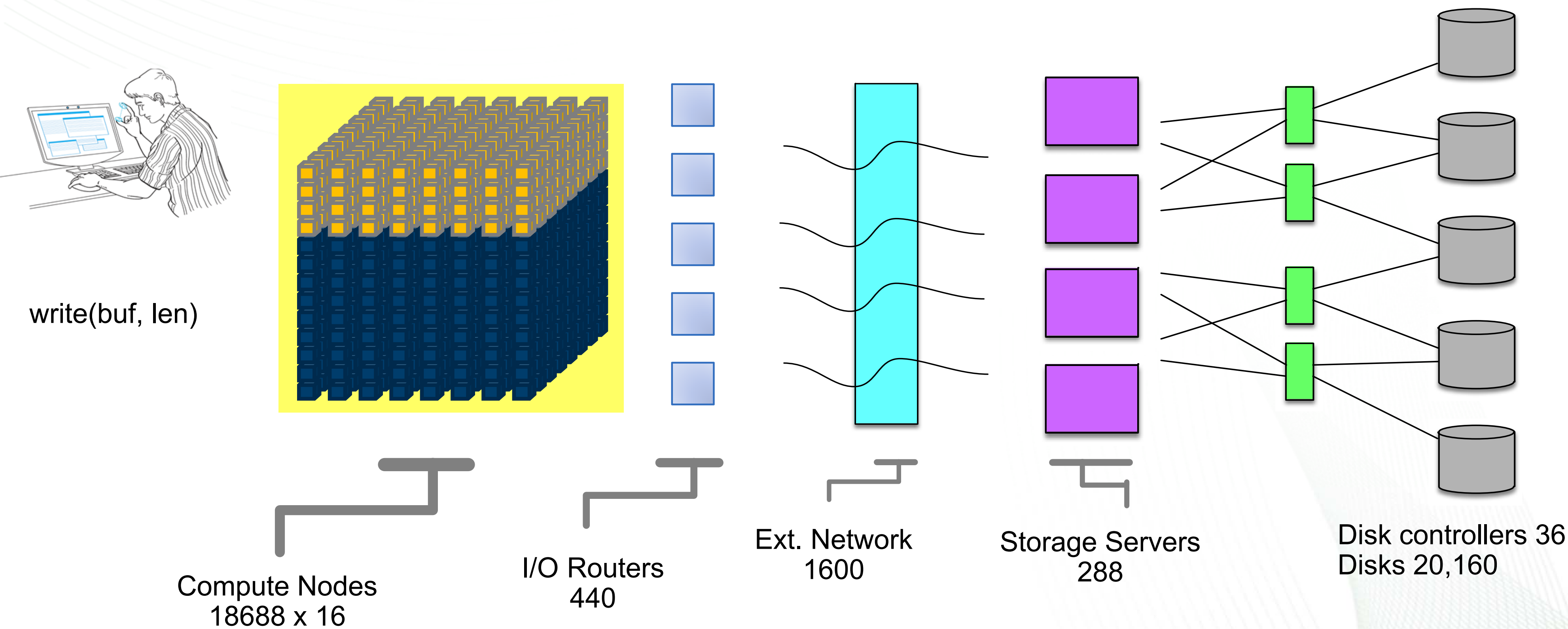


- Disadvantages
  - Possible lower performance than file per process mode due to contention

System perspective - parallel I/O

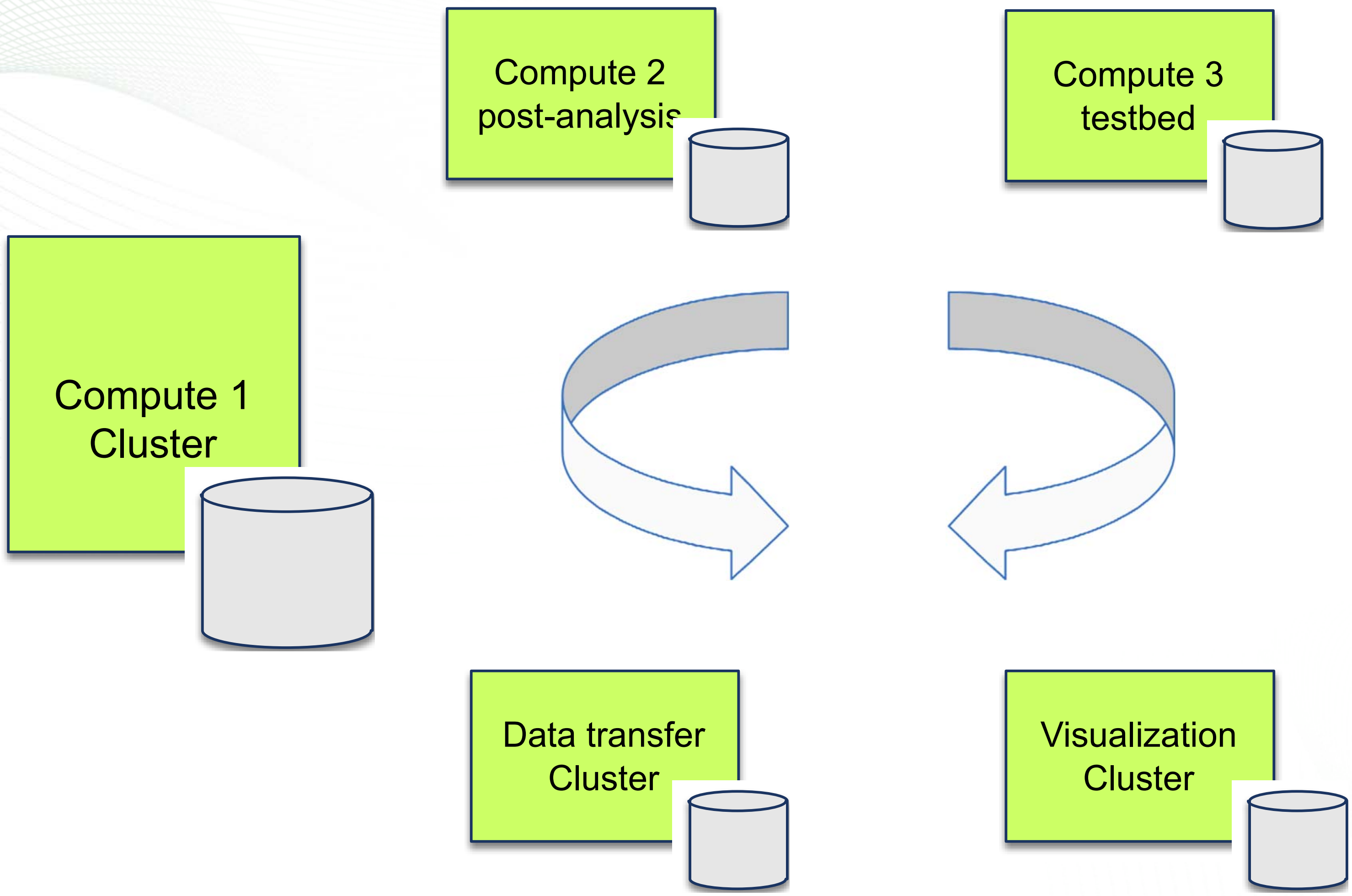
- A combination of software and hardware
- A much more complex end-to-end I/O path





How does knowing more about HPC I/O system help to understand I/O behavior and performance?

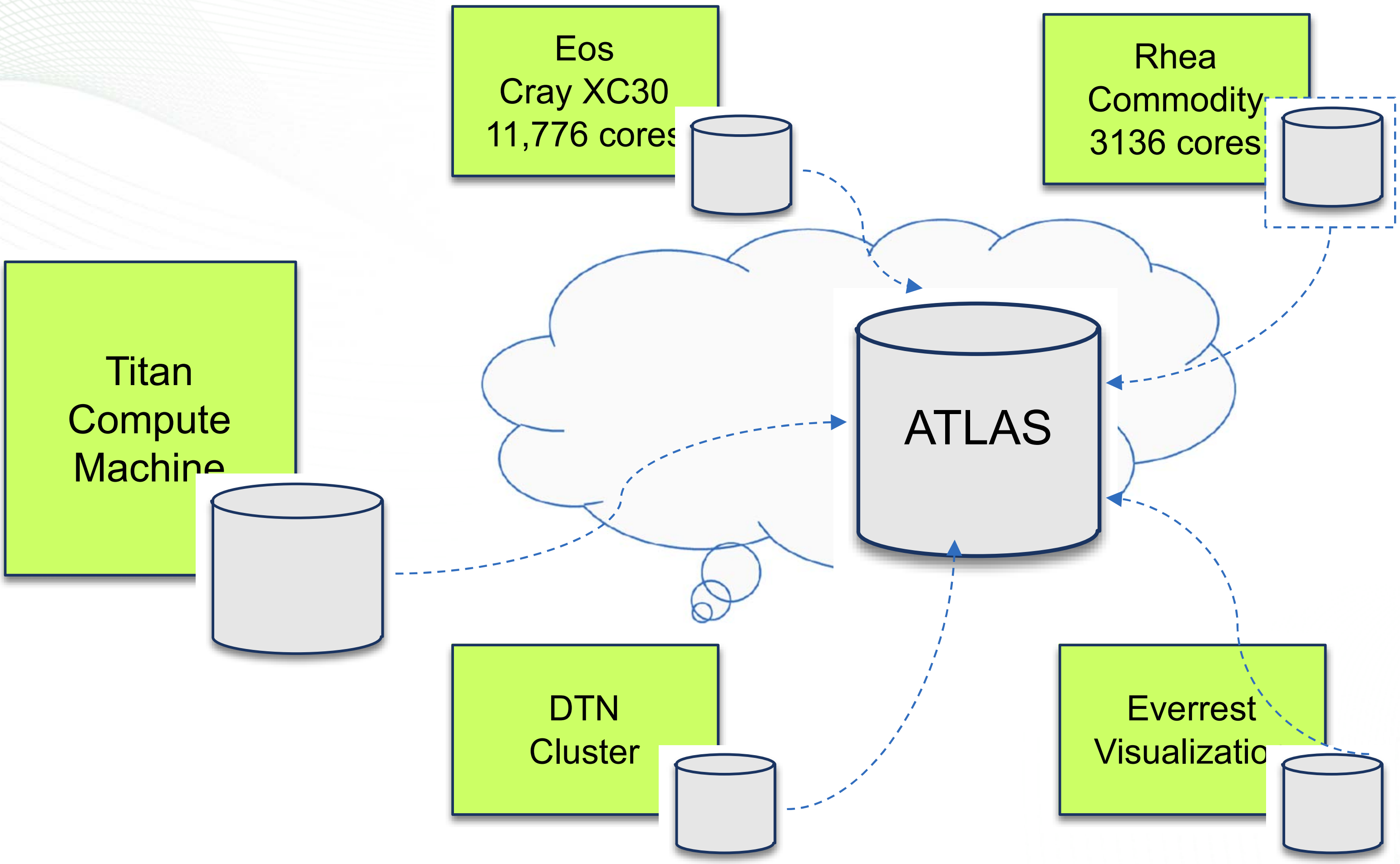
Storage Design Paradigm: Machine Exclusive



PROS: A simple and natural starting point

CONS: wasted resource, procurement, operational cost and data island

Storage Design Paradigm: Data Centric



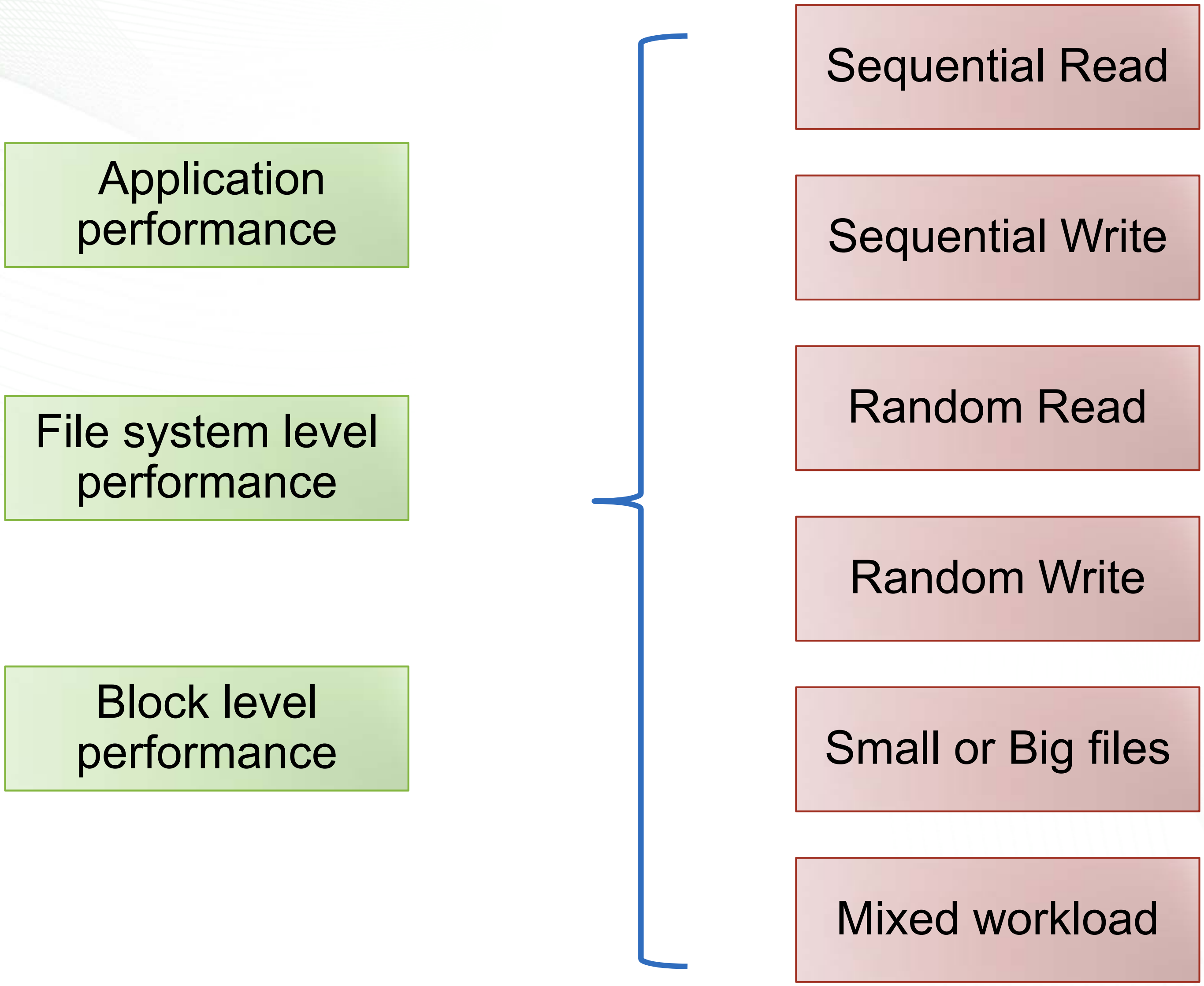
PROS: eliminate data island
& data availability

CONS: mixed workload
& data availability

System Perspective Observations

- HPC I/O system design is about trade-offs
  - performance
  - capacity
  - scalability
  - availability
  - usability
  - and cost
- Complexity and control of I/O end-to-end path
- The impact of mixed workload
  - I/O workload characterization is paramount important
  - Write is not necessarily dominant even for a scratch file system
- Performance isolation and QoS

Performance perspective: what kind?



Performance perspective: some observations

- The end-user experience (performance wise) is less predictable in HPC: latency and bandwidth
- It is extremely challenging if not impossible to achieve peak performance in production environment.
- A fraction of the whole machine can overwhelm the underlying storage system.
- The perils of hero runs:

XYZ Dilemma: *Why I am allocated X CPU cores, you have said the system is capable of performance Y, and I am only getting Z number*

OAK RIDGE National Laboratory

Some Useful Links

- NERSC I/O In Practice
  - http://www.nersc.gov/assets/Training/pio-in-practice-sc12.pdf
- Livermore Computing I/O guide
  - *https://computing.llnl.gov/LCdocs/ioguide/*
- Darshan HPC I/O Characterization Tool
  - *http://www.mcs.anl.gov/research/projects/darshan/*
  - *http://www.nersc.gov/users/software/performance-and-debugging-tools/darshan/*
- OLCF Resources
  - https://www.olcf.ornl.gov/kb_articles/spider-the-center-wide-lustre-file-system/
  - *https://www.olcf.ornl.gov/kb_articles/lustre-basics/*
  - *https://www.olcf.ornl.gov/kb_articles/darshan-basics/*

Useful I/O libraries and middleware

- ADIOS
  - https://www.olcf.ornl.gov/center-projects/adios/
- HDF5
  - https://www.hdfgroup.org/HDF5/
- NetCDF
  - http://www.unidata.ucar.edu/software/netcdf/
- Parallel NetCDF
  - https://trac.mcs.anl.gov/projects/parallel-netcdf

Summary

- Trend of scalable storage in HPC
  - More hierarchical
  - More heterogeneous
- Mapping from application data model to storage hardware efficiently is increasingly more complex
- Setting the right expectation of performance
  - Understand application I/O requirement and characterization
  - Think about data transformation and pick the right I/O interface for job
  - Measure and know the bottleneck
  - Deeper understanding on the storage architecture
  - Think about portability