ATPESC 2016

Argonne NATIONAL LABORATORY

THINKING ABOUT HPC I/O AND HPC STORAGE

ROB LATHAM
PHIL CARNS
Argonne National Laboratory

8:35-9:00am August 11, 2016
St. Charles IL

HPC I/O SYSTEMS

An HPC I/O system is a combination of hardware and software that assists in accessing and retaining data.

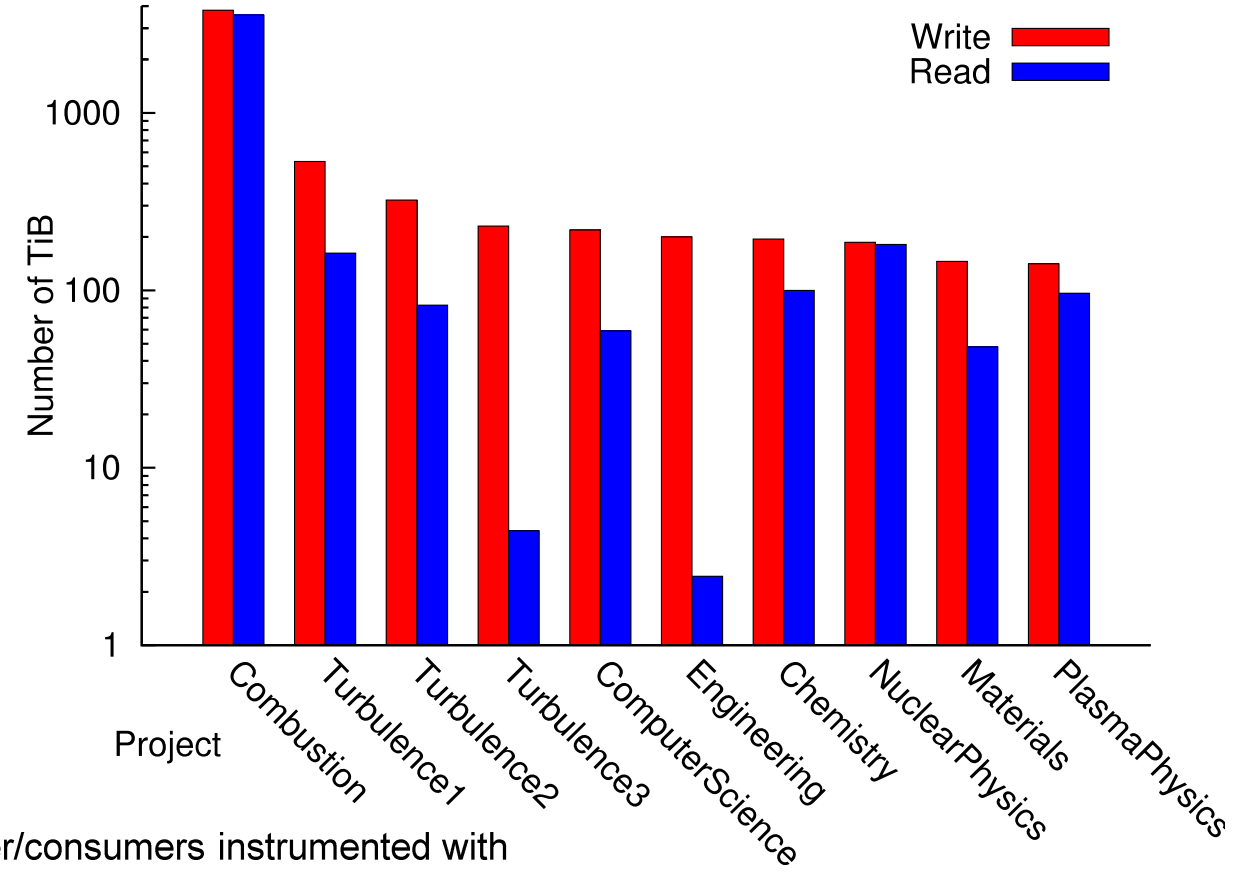
- Hardware: disks, disk enclosures, servers, networks, etc.
- Software: parallel file system, libraries, parts of the OS

I/O may be performed by applications for a variety of reasons:

- Productive I/O: storing scientific results
- Defensive I/O: saving state in case the application or system crashes
- Analysis I/O: scientific discovery from previous results

DATA VOLUMES IN COMPUTATIONAL SCIENCE

It's not just checkpoints – scientists are reading large volumes of data *into* HPC systems as part of their science.

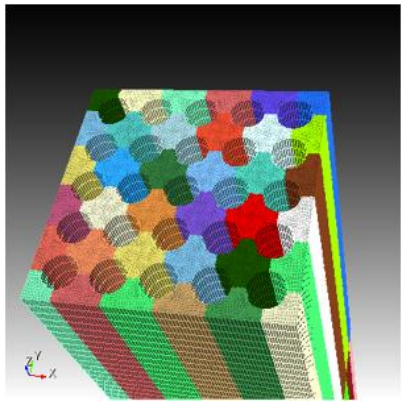


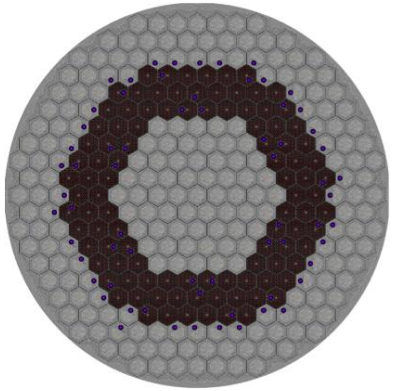
Top 10 data producer/consumers instrumented with Darshan from August 2014 to January 2015 (Mira).

DATA COMPLEXITY IN COMPUTATIONAL SCIENCE

- Applications use advanced data models to fit the problem at hand
  - Multidimensional typed arrays, images composed of scan lines, …
  - Headers, attributes on data
- I/O systems have very simple data models
  - Tree-based hierarchy of containers
  - Some containers have streams of bytes (files)
  - Others hold collections of other containers (directories or folders)

Effective mapping from application data models to I/O system data models is the key to I/O performance.





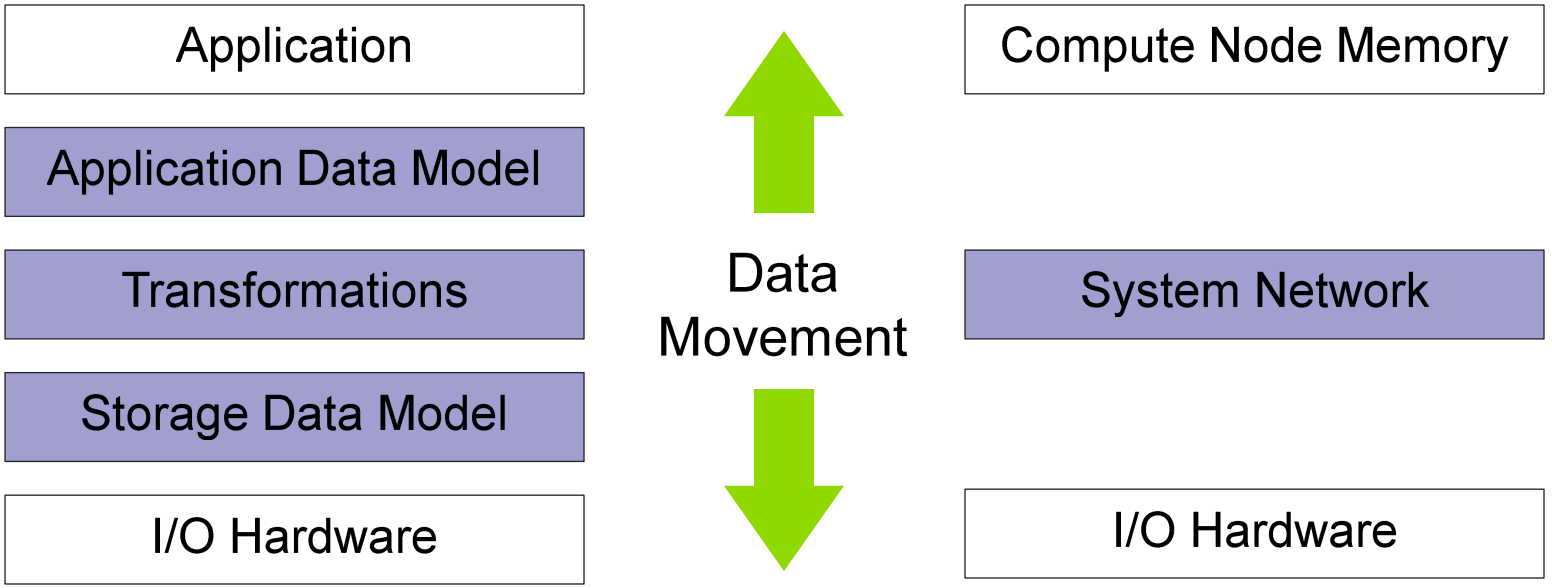
Model complexity: Spectral element mesh (top) for thermal hydraulics computation coupled with finite element mesh (bottom) for neutronics calculation.

Scale complexity: Spatial range from the reactor core in meters to fuel pellets in millimeters.

Images from T. Tautges (ANL) (upper left), M. Smith (ANL) (lower left), and K. Smith (MIT) (right).

VIEWS OF DATA ACCESS IN HPC SYSTEMS

Two useful ways of thinking about data access are the "logical" view, considering data models in use, and the "physical" view, the components that data resides on and passes through.

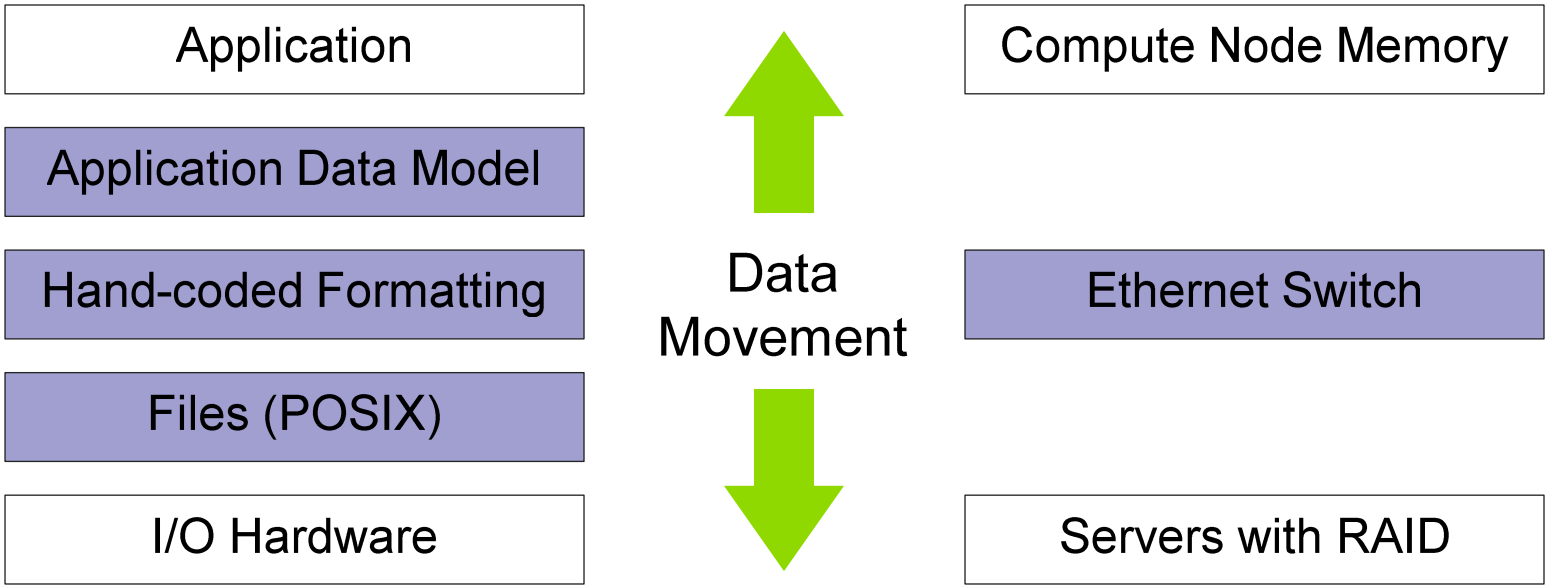


Logical (data model) view of data access.

Physical (hardware) view of data access.

DATA ACCESS IN PAST HPC SYSTEMS*

For many years, application teams wrote their own translations from their data models into files, and hardware model was relatively simple.



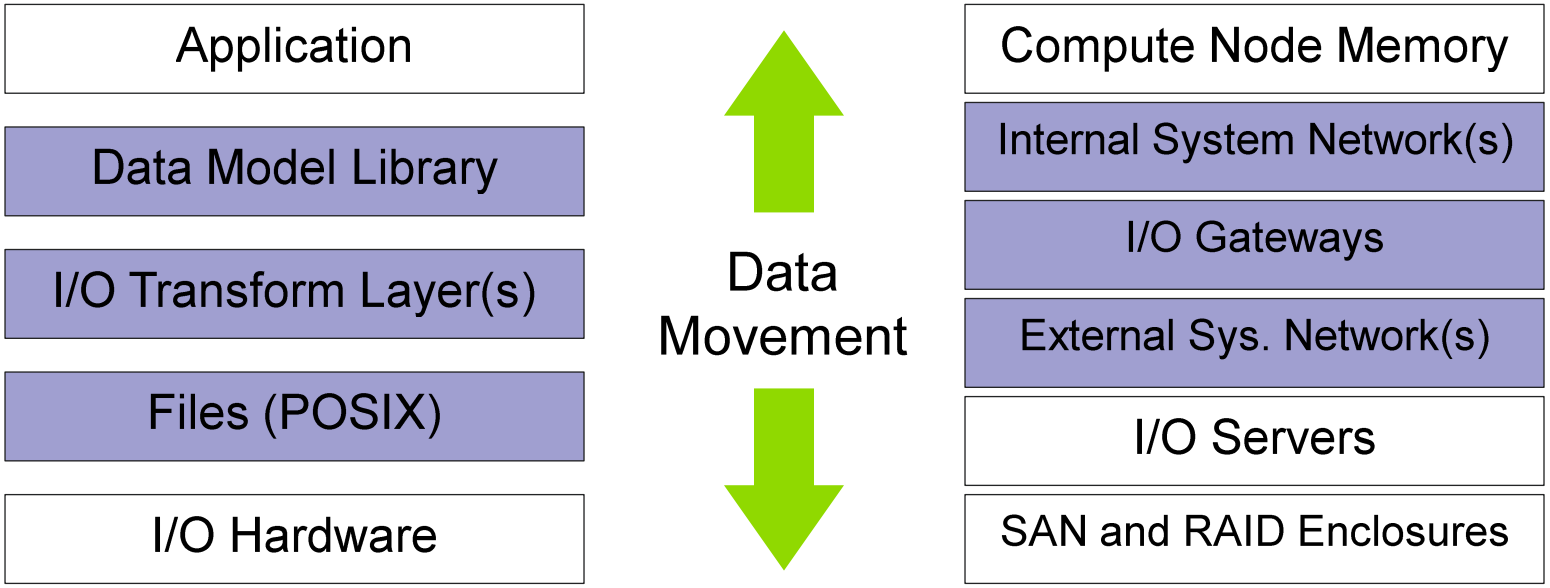
Logical (data model) view of data access.

Physical (hardware) view of data access.

* We're simplifying the story here somewhat …

DATA ACCESS IN CURRENT LARGE-SCALE SYSTEMS

Current systems have greater support on the logical side, more complexity on the physical side.



Logical (data model) view of data access.

Physical (hardware) view of data access.

- Does this mean that applications must be more complex as well? *No!*
- More responsibility is assumed by system software and hardware in this model. It's just that there are more components to be aware of.

HPC I/O SOFTWARE STACK

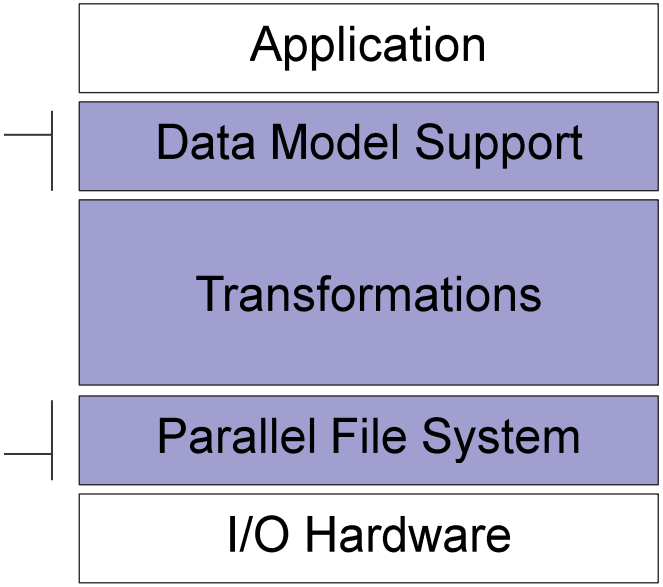
The software used to provide data model support and to transform I/O to better perform on today's I/O systems is often referred to as the *I/O stack.*

Data Model Libraries map application abstractions onto storage abstractions and provide data portability.

HDF5, Parallel netCDF, ADIOS

Parallel file system maintains logical file model and provides efficient access to data.

PVFS, PanFS, GPFS, Lustre



I/O Middleware organizes accesses from many processes, especially those using collective I/O.

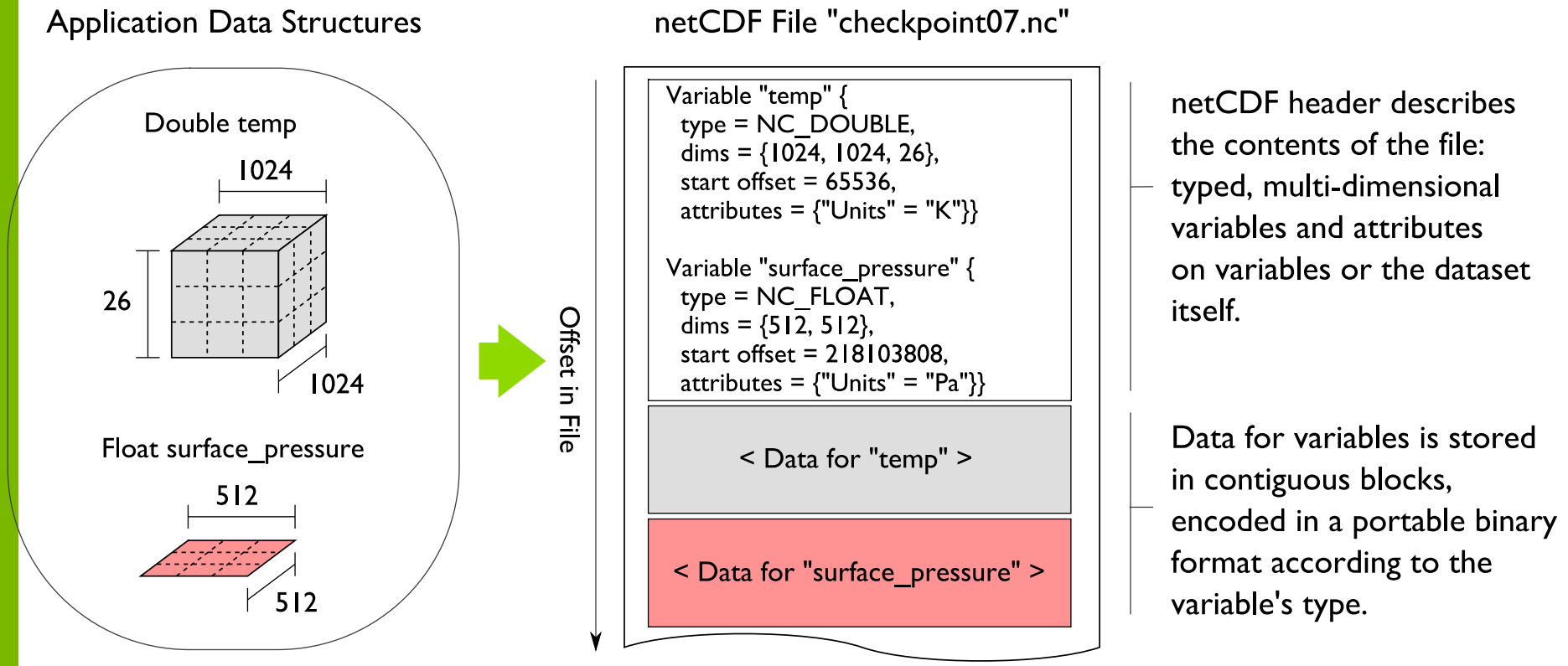
MPI-IO, GLEAN, PLFS

I/O Forwarding transforms I/O from many clients into fewer, larger request; reduces lock contention; and bridges between the HPC system and external storage.

IBM ciod, IOFSL, Cray DVS, Cray Datawarp

EXAMPLE OF ORGANIZING APPLICATION DATA

Application data models are supported via libraries that map down to files (and sometimes directories).



THINKING ABOUT HPC I/O SYSTEMS

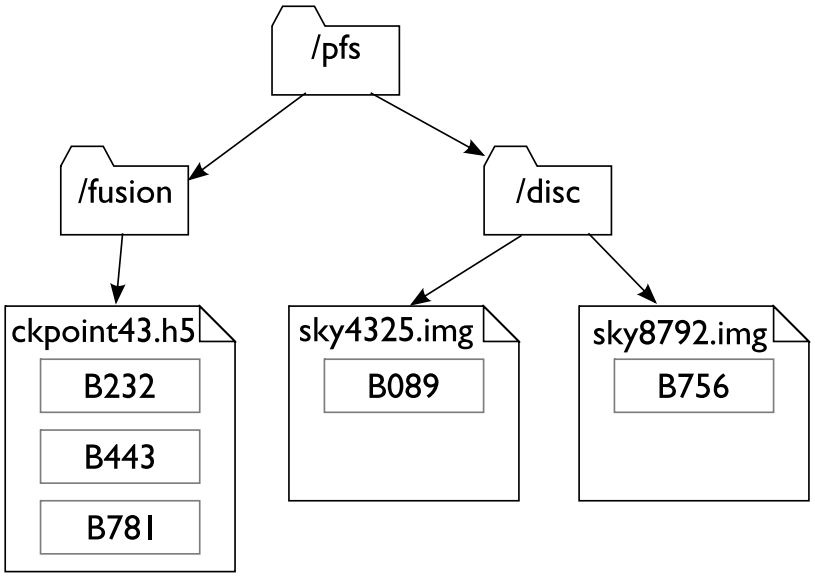
- Next we will summarize some internal details of HPC I/O systems that will help to provide context for the optimizations and best practices that we will talk about later in the day
- Lots of computer science R&D has gone into making these systems faster and easier to use
- Our objective for the day is to share the best practices and tools that have arisen from this R&D effort

HOW IT WORKS: HPC I/O SYSTEMS

HOW IT WORKS

- HPC I/O systems provide a *file system view* of stored data
  - File and directory model
  - Coherent view of data across the system
  - Access from compute nodes, login nodes, and grid transfers

- Topics:
  - How is data stored and organized?
  - What support is there for application data models?
  - How does data move from clients to servers?
  - How is concurrent access managed?
  - What transformations are typically applied?



File system view consists of directories (a.k.a. folders) and files. Files are broken up into regions called extents or blocks.

STORING AND ORGANIZING DATA: STORAGE MODEL

HPC I/O systems are built around *parallel file systems* that organizes storage and manages access.

- Parallel file systems (PFSes) are distributed systems that provide a file data model (i.e., files and directories) to users
- Multiple PFS servers manage access to storage, while PFS client systems run applications that access storage
- PFS clients can access storage resources in parallel
- On the surface it looks just like any other file system (home directory, laptop, etc.) but with different performance properties.

WHAT TO EXPECT FROM HPC I/O SYSTEMS

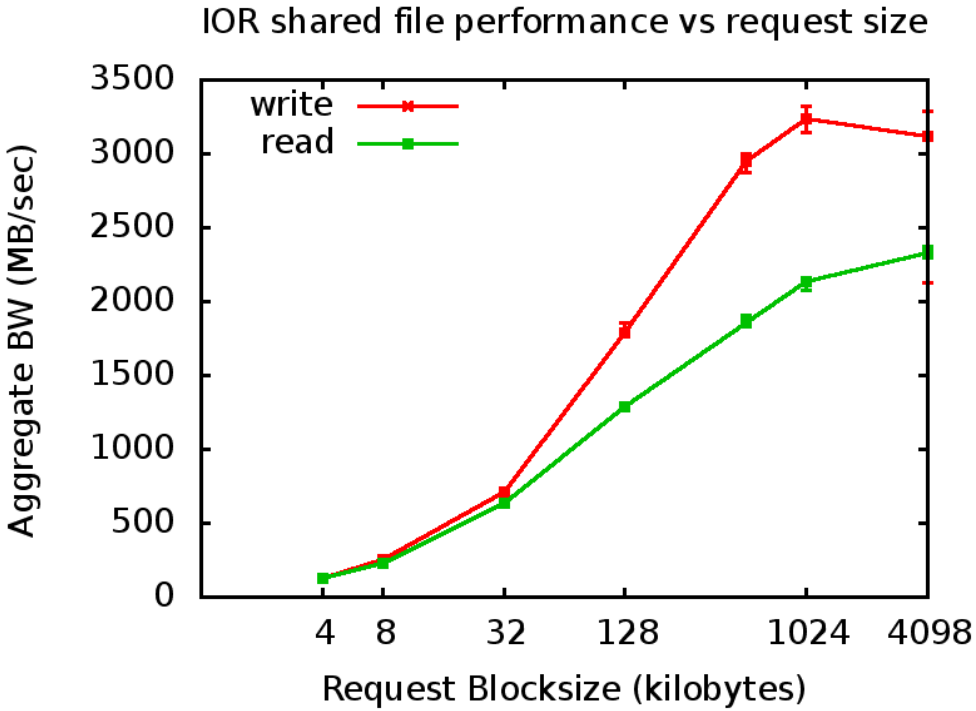
In general, how do large-scale HPC I/O systems differ from conventional file systems (on servers, clusters, or even your laptop)?

- Vastly more bandwidth
  - There are a lot of disks and network links that can be used simultaneously
  - But you ***must*** read/write in parallel to exploit it
- Vastly worse latency
  - Multiple network/bus hops to get from application to disk

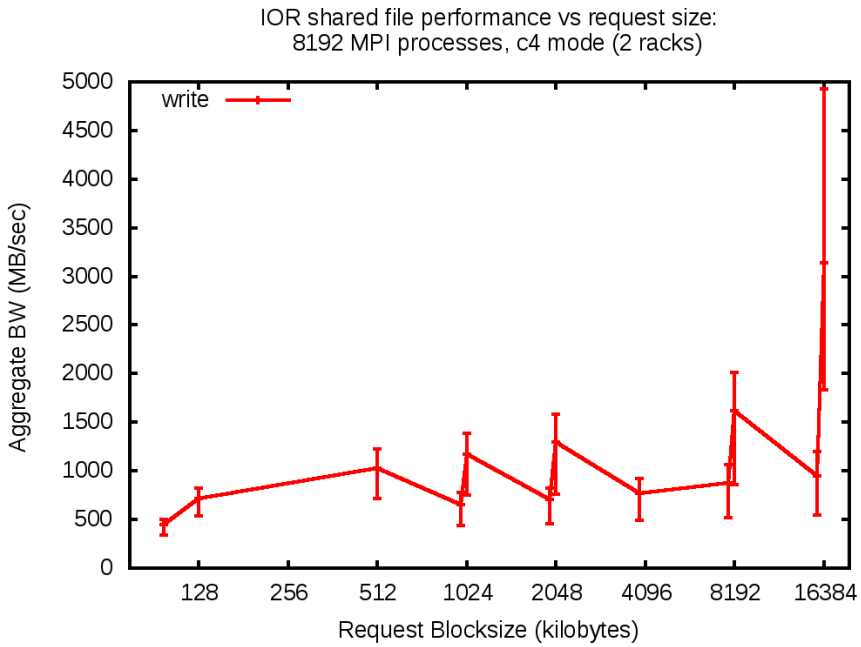
Most of the optimizations discussed in this presentation revolve around a central theme: organizing your data so that you can take advantage of the bandwidth while avoiding the latency.

THE IMPACT OF REQUEST SIZES

Interconnect latency has a significant impact on effective rate of I/O. Typically I/Os should be in the O(Mbytes) range.



2K processes of IBM Blue Gene/P at ANL.



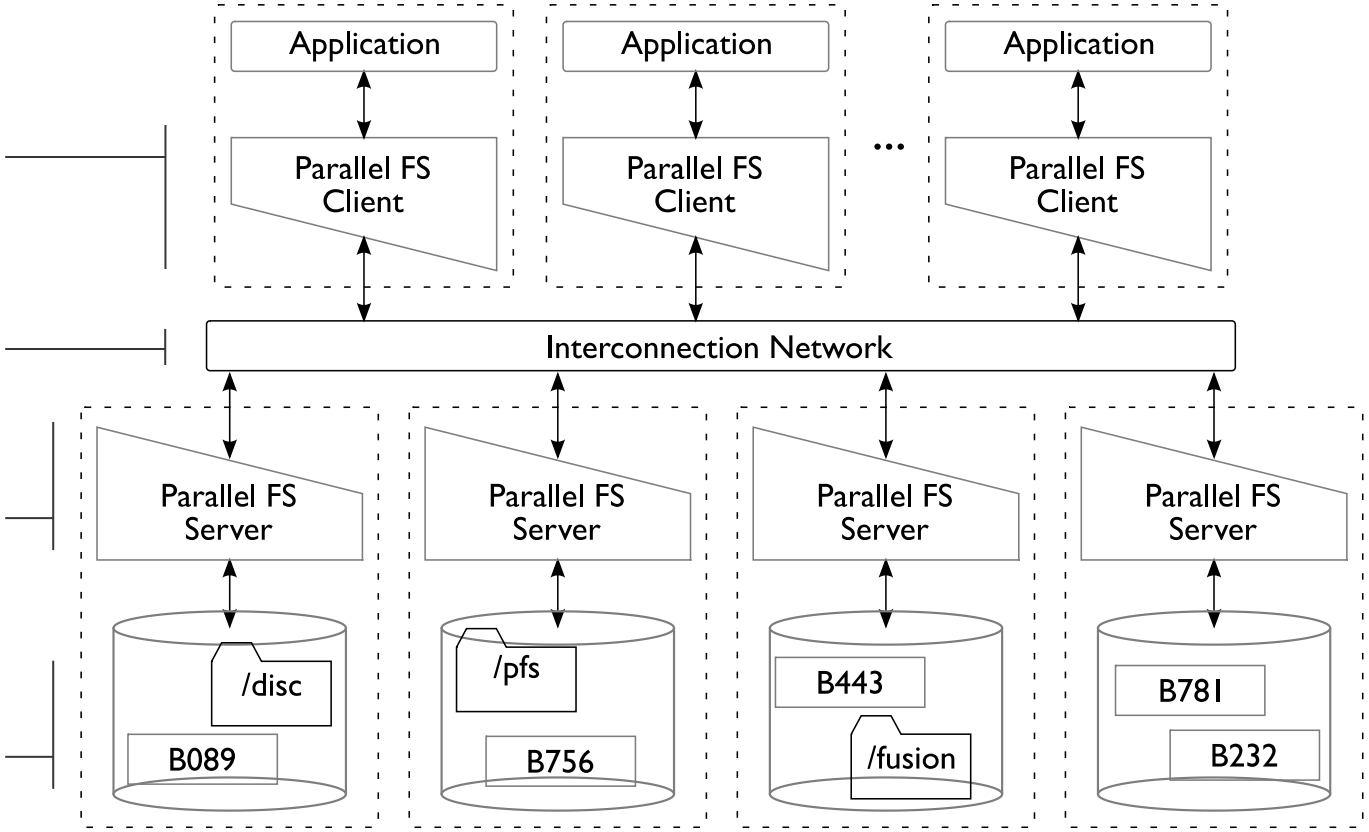
8k processes of IBM Blue Gene /Q at ANL

READING AND WRITING DATA (ETC.)

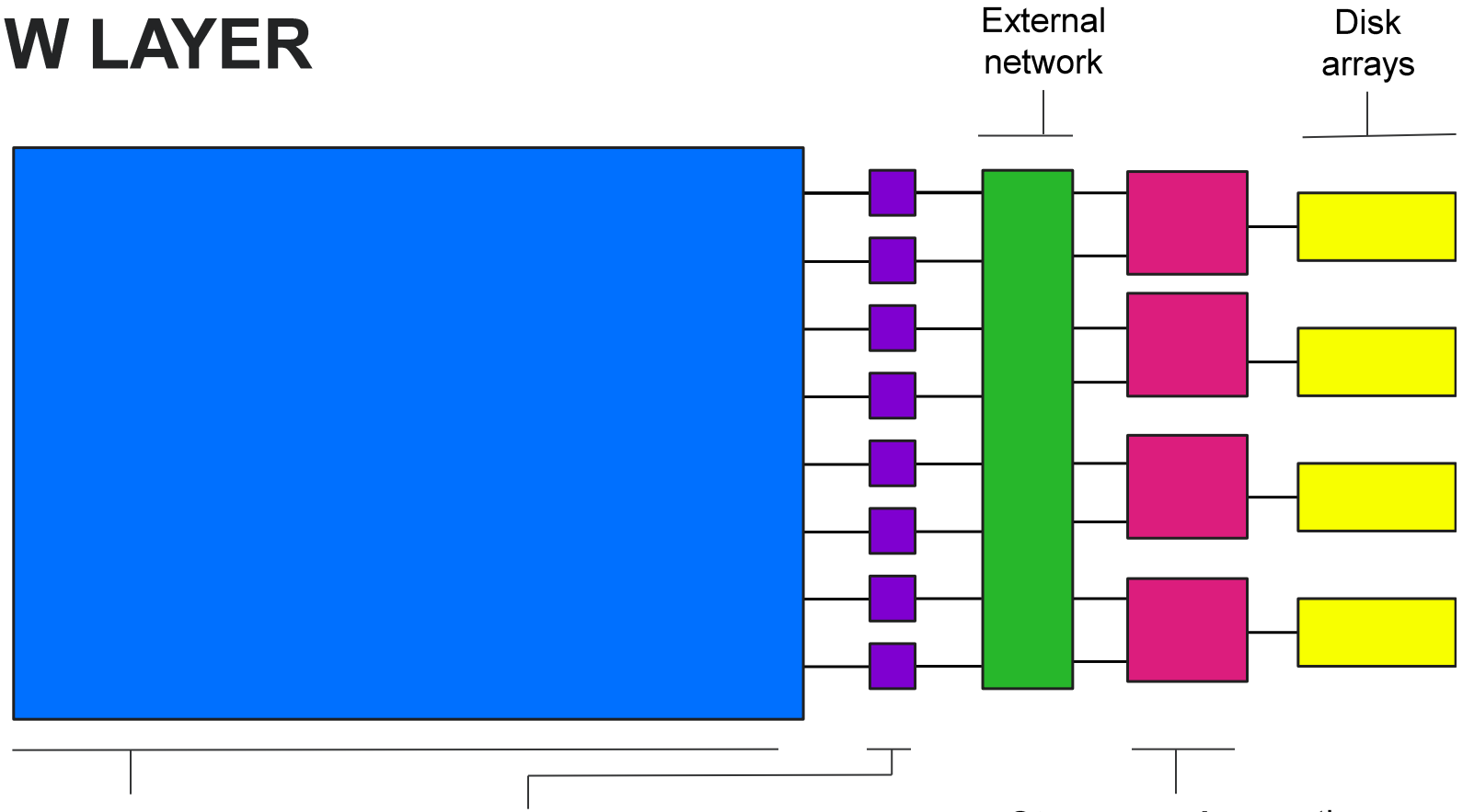
PFS client software requests operations on behalf of applications. Requests are sent as messages (RPC-like), often to multiple servers. Requests pass over the interconnect, thus **each request incurs some latency**.

PFS servers manage local storage, services incoming requests from clients.

RAID enclosures protect against individual disk failures and map regions of data onto specific devices.



LEADERSHIP SYSTEMS HAVE AN ADDITIONAL HW LAYER



Compute nodes run application processes. Data model software also runs here, and some I/O transformations are performed here.

I/O forwarding nodes (or I/O gateways) shuffle data between compute nodes and external resources, including storage.

Storage nodes run the parallel file system.

Note: systems coming online now that include burst buffers in the forwarding layer or on the compute nodes themselves. We will talk about that in a later session.

HPC PLATFORMS: IT'S A JUNGLE OUT THERE

- HPC systems are purpose-built by a handful of different vendors
- Their storage systems are no different. Major storage platforms in the DOE include:
  - GPFS (IBM)
  - Lustre (Intel)
  - PanFS (Panasas)

- …In some cases with different hardware integrators, and almost always with different performance characteristics
- *Storage systems are complex, storage systems are not performance portable, we have our work cut out for us!*
- Today's session will focus on solutions to this problem:
  - Libraries that take responsibility for performance portability
  - Instrumentation tools to understand I/O behavior
  - Cross-site data transfer methods
  - Best practices that apply to all HPC systems

THANK YOU!

THIS CONCLUDES THE “THINKING ABOUT HPC I/O AND HPC STORAGE” PRESENTATION

ON DECK: HPC TRANSFORMATIONS