Argonne NATIONAL LABORATORY

Trends in HPC I/O and File Systems

Rob Ross, Phil Carns, Dave Goodell, Kevin Harms, Kamil Iskra, Dries Kimpe, Rob Latham, Tom Peterka, Rajeev Thakur, and Venkat Vishwanath

Mathematics and Computer Science Division

Argonne National Laboratory

rross@mcs.anl.gov

U.S. DEPARTMENT OF ENERGY

Exascale Systems: I/O Gap

Systems	2009	2018*	Difference
System Peak	2 Pflop/sec	1 Eflop/sec	O(1000)
Power	6 Mwatt	20 Mwatt	
System Memory	0.3 Pbytes	32-64 Pbytes	O(100)
Node Compute	125 Gflop/sec	1-15 Tflop/sec	O(10-100)
Node Memory BW	25 Gbytes/sec	2-4 Tbytes/sec	O(100)
Node Concurrency	12	O(1-10K)	O(100-1000)
Total Node Interconnect BW	3.5 Gbytes/sec	200-400 Gbytes/sec	O(100)
System Size (Nodes)	18,700	O(100,000-1M)	O(10-100)
Total Concurrency	225,000	O(1 billion)	O(10,000)
Storage	**15 Pbytes**	**500-1000 Pbytes**	**O(10-100)**
I/O	**0.2 Tbytes/sec**	**60 Tbytes/sec**	**O(100)**
MTTI	Days	O(1 day)	

From J. Dongarra, "Impact of Architecture and Technology for Extreme Scale on Software and Algorithm Design," Cross-cutting Technologies for Computing at the Exascale, February 2-5, 2010.

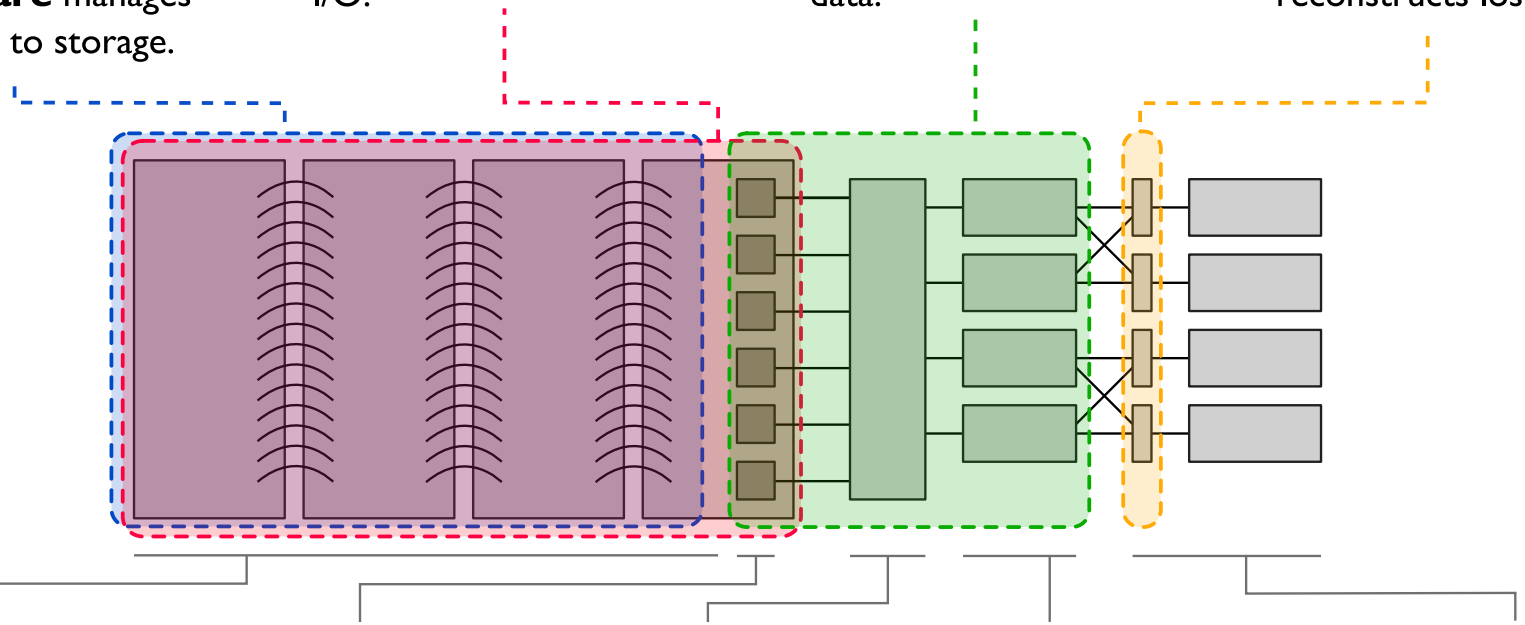
Production I/O Software Stack and HW Architecture

High-level I/O libraries execute on compute nodes, mapping application abstractions into flat files, and encoding data in portable formats.
I/O middleware manages collective access to storage.

I/O forwarding software runs on compute and gateway nodes, bridges networks, and provides aggregation of independent I/O.

Parallel file system code runs on gateway and storage nodes, maintains logical storage space and enables efficient access to data.

Drive management software or firmware executes on storage controllers, organizes individual drives, detects drive failures, and reconstructs lost data.



Compute nodes
40,960 Quad core PowerPC 450 nodes with 2 Gbytes of RAM each

Gateway nodes
640 Quad core PowerPC 450 nodes with 2 Gbytes of RAM each

Commodity network
900+ port 10 Gigabit Ethernet Myricom switch complex

Storage nodes
136 two dual core Opteron servers with 8 Gbytes of RAM each

Enterprise storage
17 DataDirect S2A9900 controller pairs with 480 1 Tbyte drives and 8 InfiniBand ports per pair

Architectural diagram of the 557 TFlop IBM Blue Gene/P system at the Argonne Leadership Computing Facility.

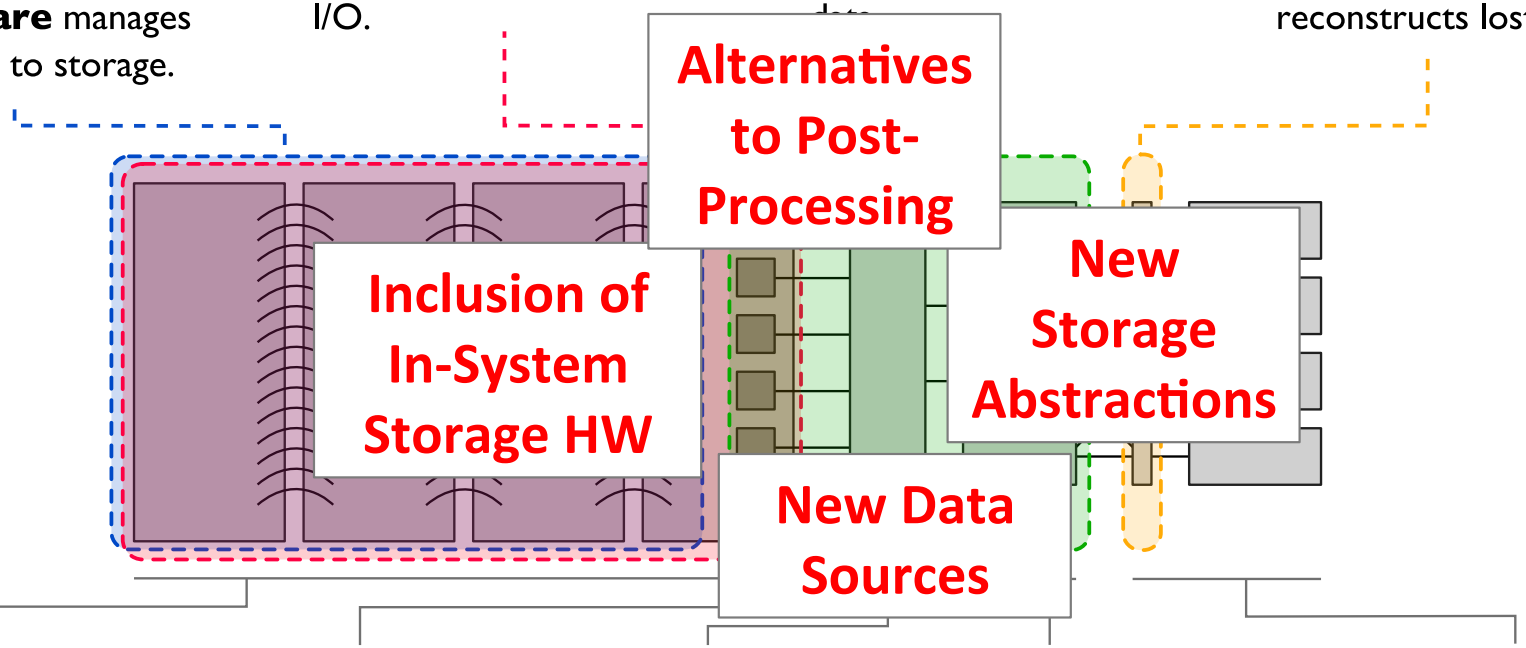
Production I/O Software Stack and HW Architecture

High-level I/O libraries execute on compute nodes, mapping application abstractions into flat files, and encoding data in portable formats.
I/O middleware manages collective access to storage.

I/O forwarding software runs on compute and gateway nodes, bridges networks, and provides aggregation of independent I/O.

Parallel file system code runs on gateway and storage nodes, maintains logical storage space and enables efficient access to

Drive management software or firmware executes on storage controllers, organizes individual drives, detects drive failures, and reconstructs lost data.



Compute nodes
40,960 Quad core PowerPC 450 nodes with 2 Gbytes of RAM each

Gateway nodes
640 Quad core PowerPC 450 nodes with 2 Gbytes of RAM each

Commodity network
900+ port 10 Gigabit Ethernet Myricom switch complex

Storage nodes
136 two dual core Opteron servers with 8 Gbytes of RAM each

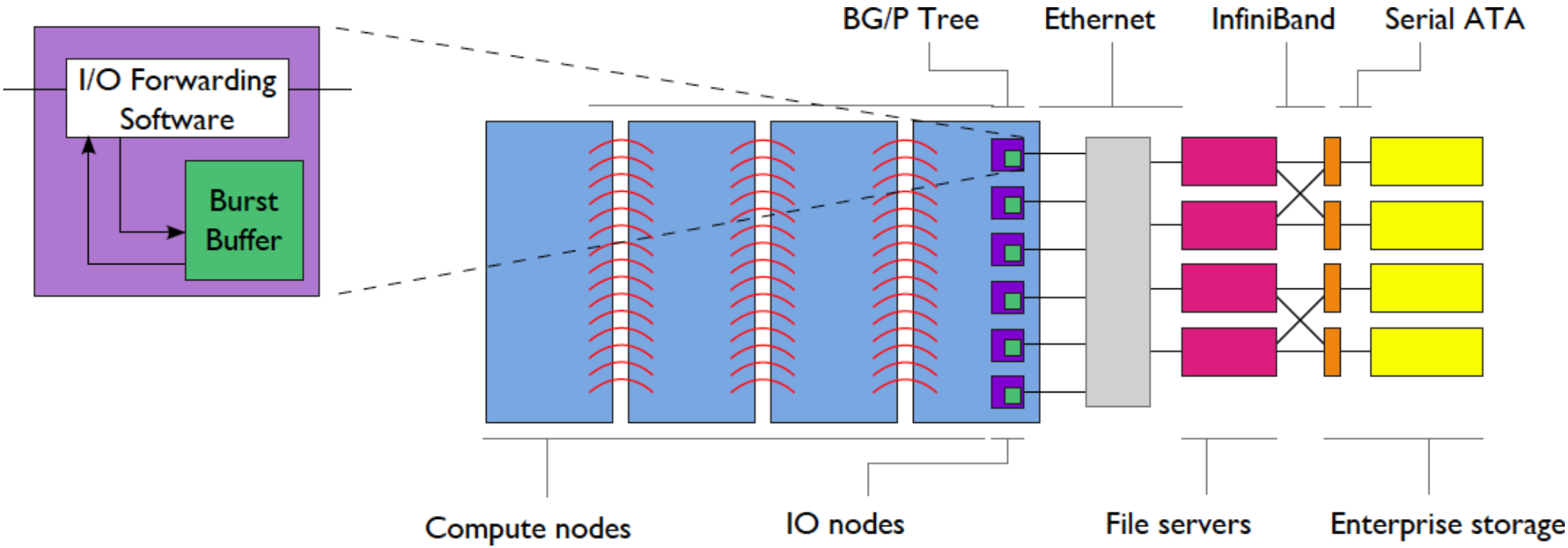
Enterprise storage
17 DataDirect S2A9900 controller pairs with 480 1 Tbyte drives and 8 InfiniBand ports per pair

Architectural diagram of the 557 TFlop IBM Blue Gene/P system at the Argonne Leadership Computing Facility.

In-System Storage

Adding Burst Buffers to the Storage Model

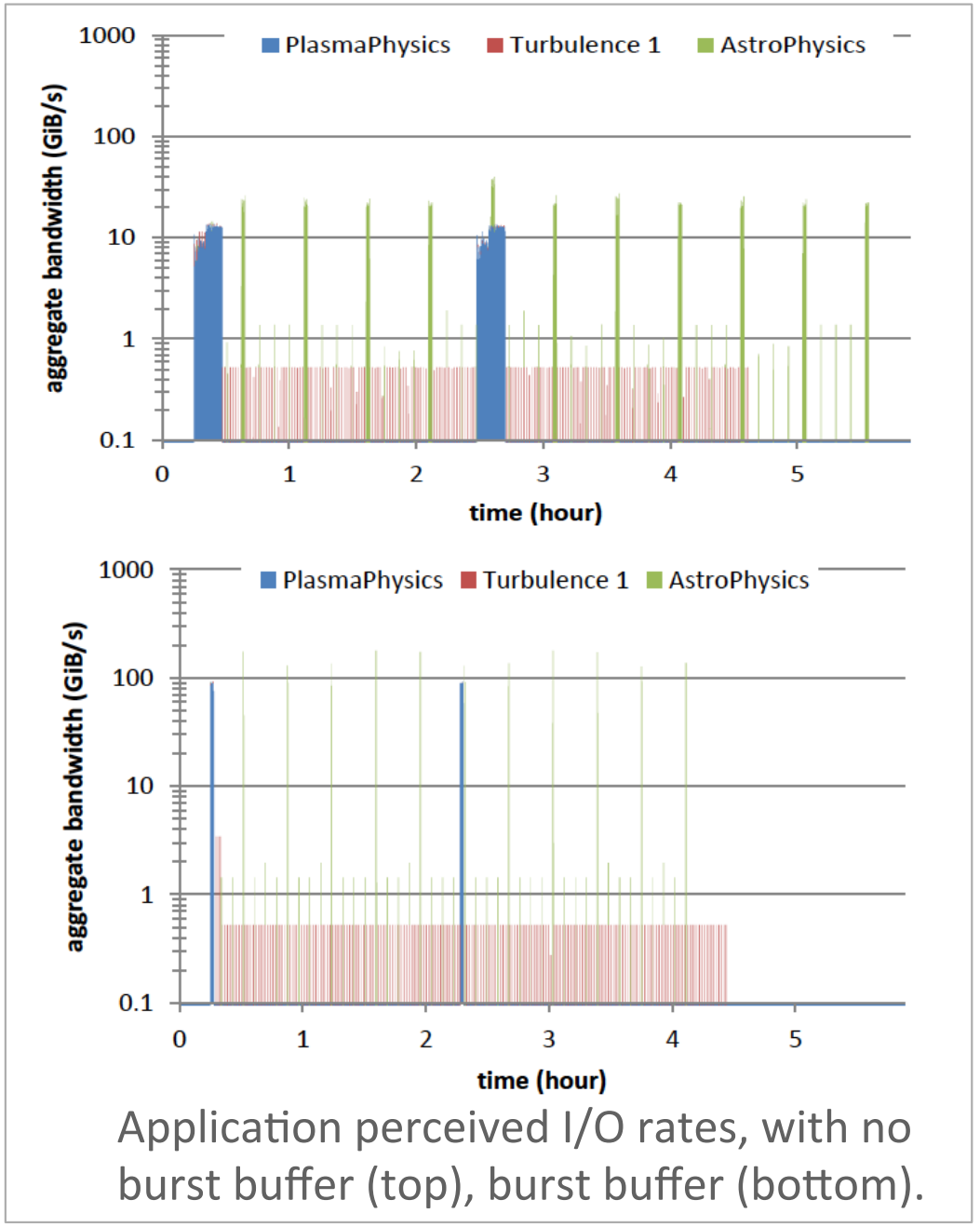
The inclusion of NVRAM storage in future systems is a compelling way to deal with the burstiness of I/O in HPC systems, reducing the peak I/O requirements for external storage. CODES can simulate this architecture before systems are built.



N. Liu et al. On the role of burst buffers in leadership-class storage systems. In Proceedings of the 2012 IEEE Conference on Massive Data Storage (MSST), Pacific Grove, CA, April 2012.

Burst Buffers Work for Multi-Application Workloads

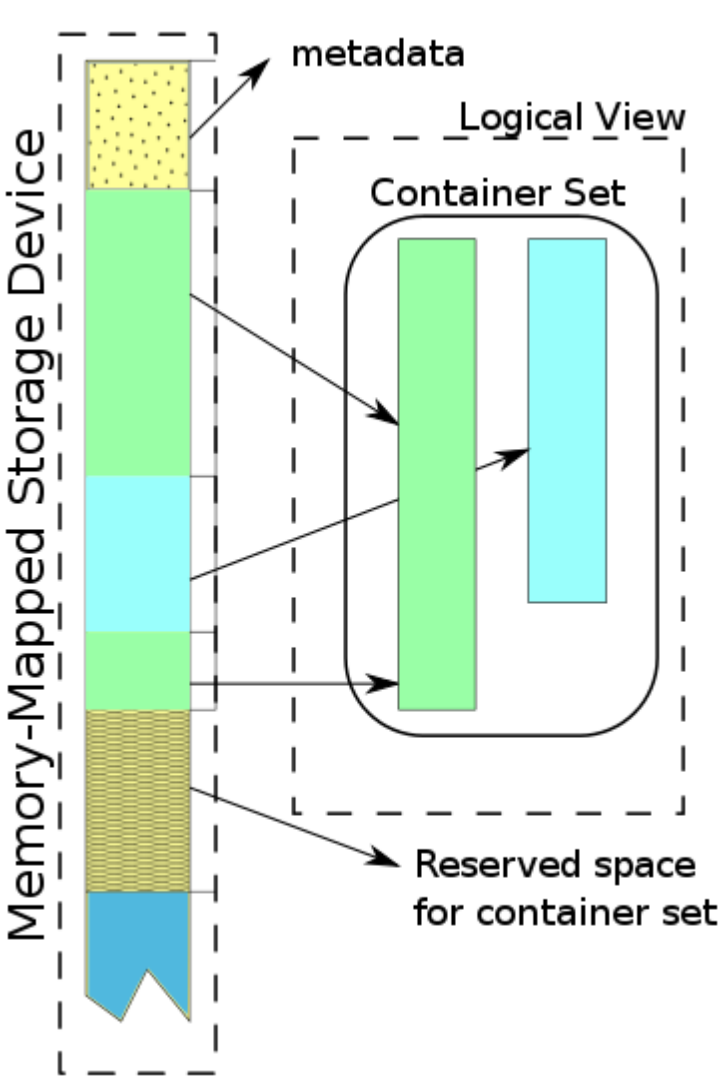
- **First detailed simulations of burst buffer behavior in a leadership computing context**
- Burst buffers accelerate the application perceived throughput under mixed I/O workloads.
- Applications' time to solution decrease with burst buffers enabled (from 5.5 to 4.4 hours)
- I/O workload language allows us to explore interplay of multiple applications performing I/O
- Peak bandwidth of the external I/O system may be reduced by 50% without a perceived change on the application side
- Tool for co-design



Application perceived I/O rates, with no burst buffer (top), burst buffer (bottom).

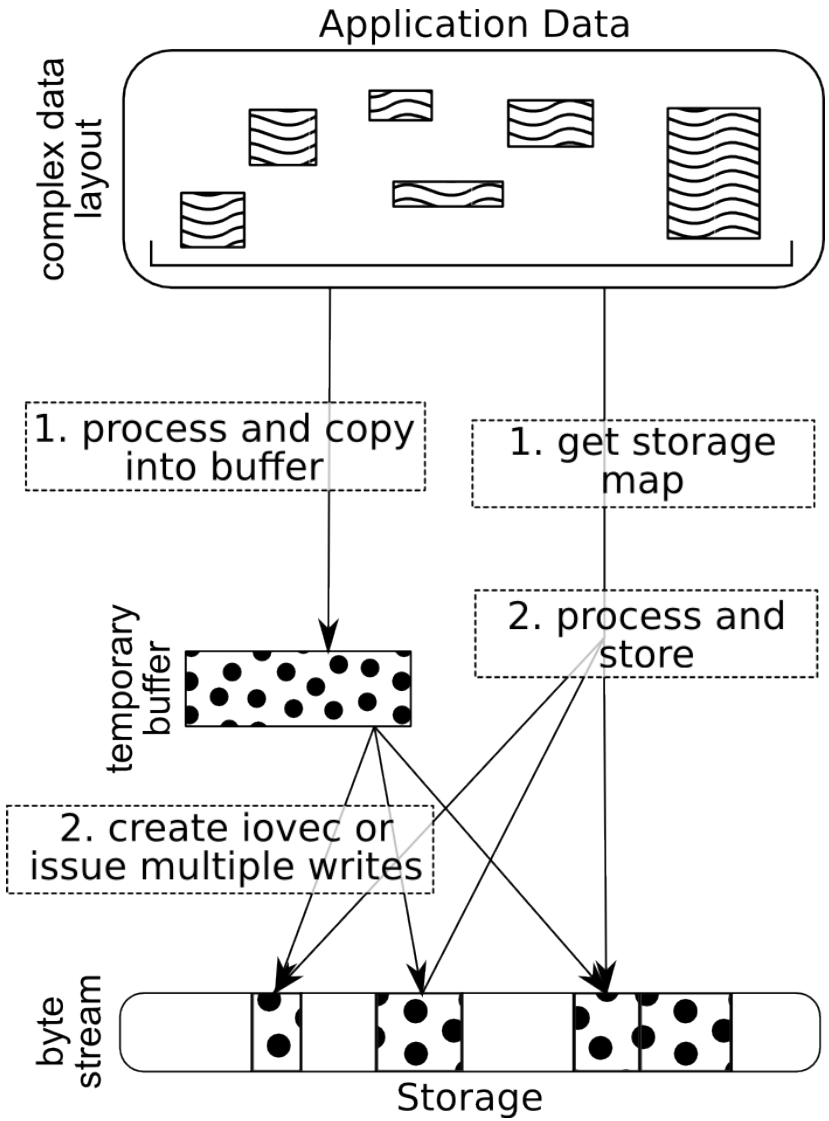
Organizing In-System Storage: Container Abstraction

- Data model for in-system storage
  - designed for **memory mapped** storage
- Why not a file on a local file system?
  - Functionality not needed for local temporary storage can come at a high cost
  - Hard to add new capabilities
  - Requires admin. privileges
- User-space implementation simplifies **experimentation** and **deployment**
- API overview:
  - `create`, `delete`, `rename`, get attributes, non-contiguous `read`, `write`, and **explicit** `resize`.
  - Each container has a name
  - Containers grouped in sets for isolation, space reservation



Direct Storage Access

- Expose container storage layout
  - Storage format designed for direct access
  - **Application transfers** data
    - Avoid extra copy (processing data)
    - No complicated non-contiguous I/O description needed.
- Compare:
  - memory-mapped I/O (extra copy)
  - XIP (no write support, FS dependent)
  - direct-io (alignment restrictions, API bottleneck)
- Layout returned as set of pointers into storage

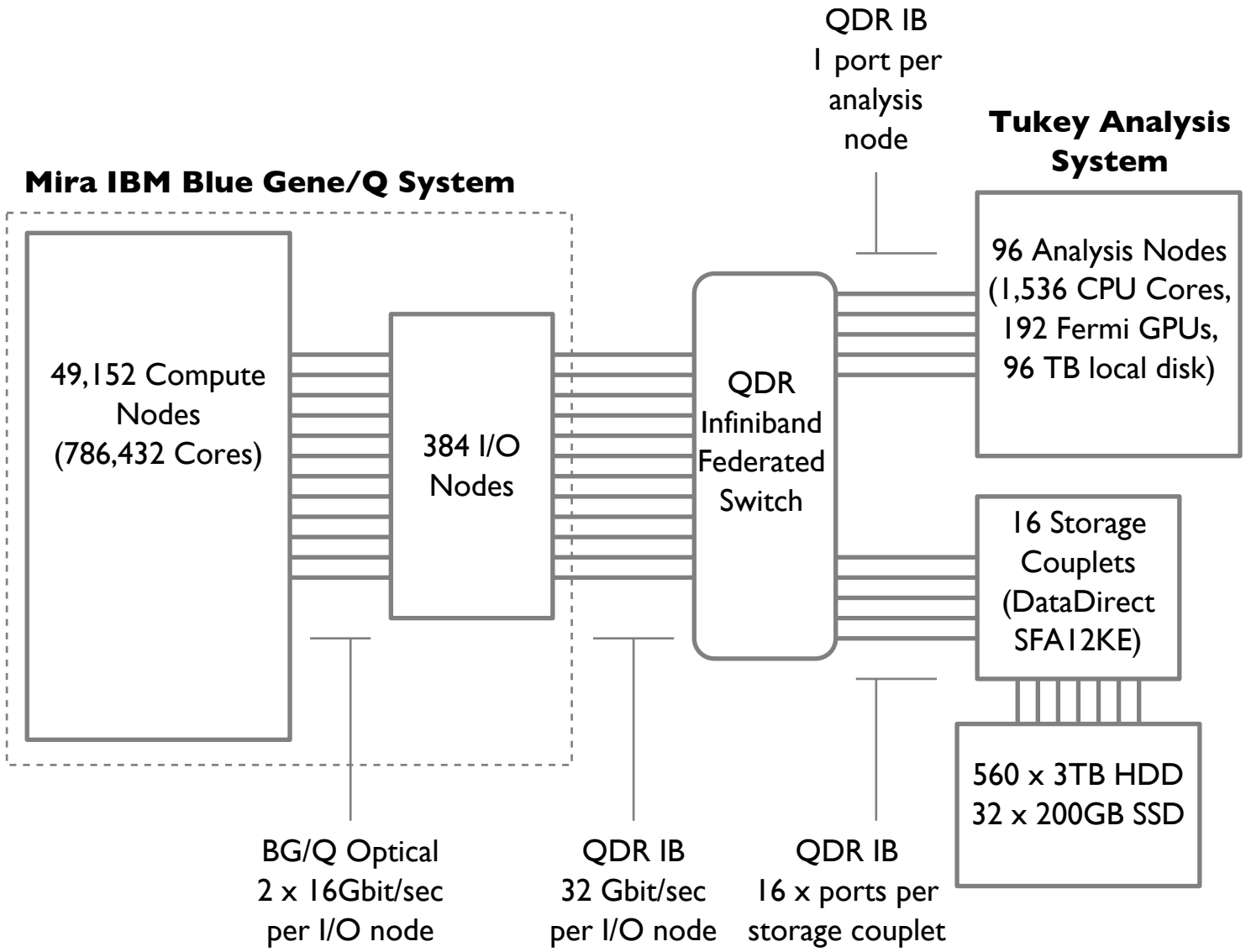


Other Work

- Scalable Checkpoint Restart (SCR)
  - Using available in-system storage to hold checkpoint data, rather than writing to external storage system
  - Working with us to develop and use container API
  - A. Moody et al. Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System. SC10, November, 2010.
- EMC PLFS Burst Buffer
  - Prototyping boxes with NVRAM to attach between HPC system and storage system
  - Using existing I/O forwarding software (IOFSL) from prior collaborative project (in some cases)
  - J. Bent et al. Jitter-Free Co-Processing on a Prototype Exascale Storage Stack. MSST 2012. April, 2012.

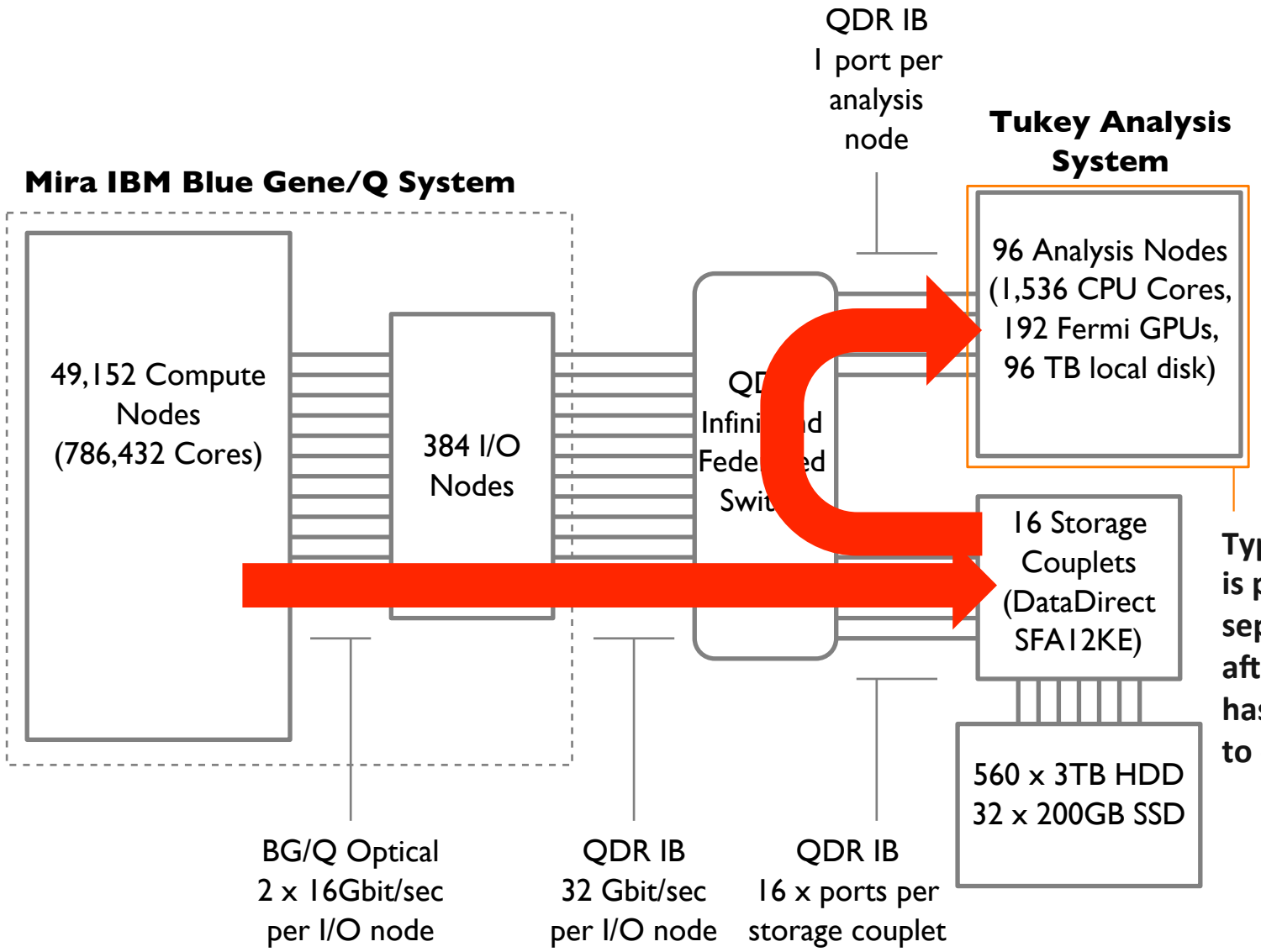
Alternatives to Post-Processing

System Architecture, Including Analysis Resources



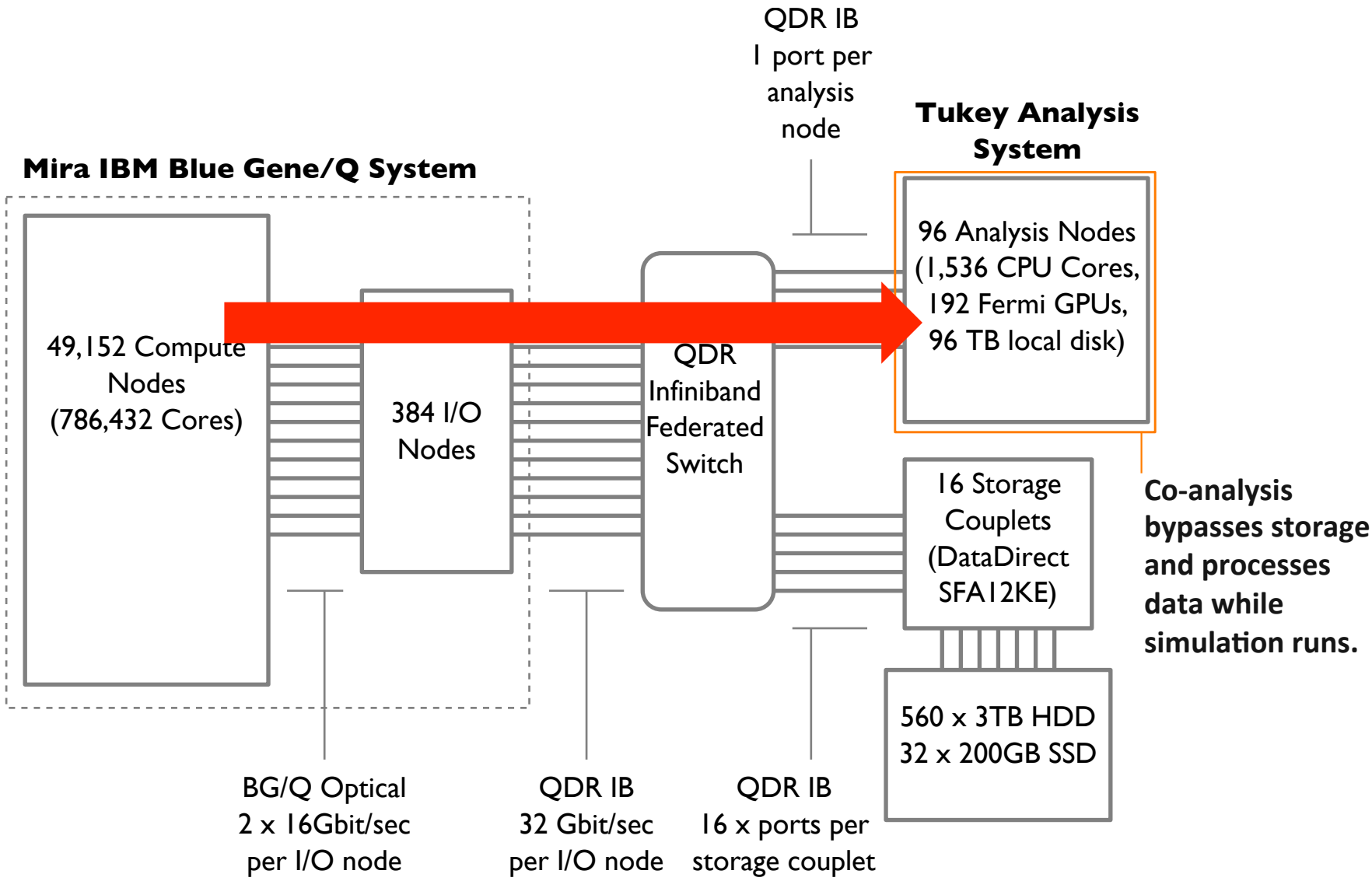
High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

Analyzing Data: Traditional Post-Processing



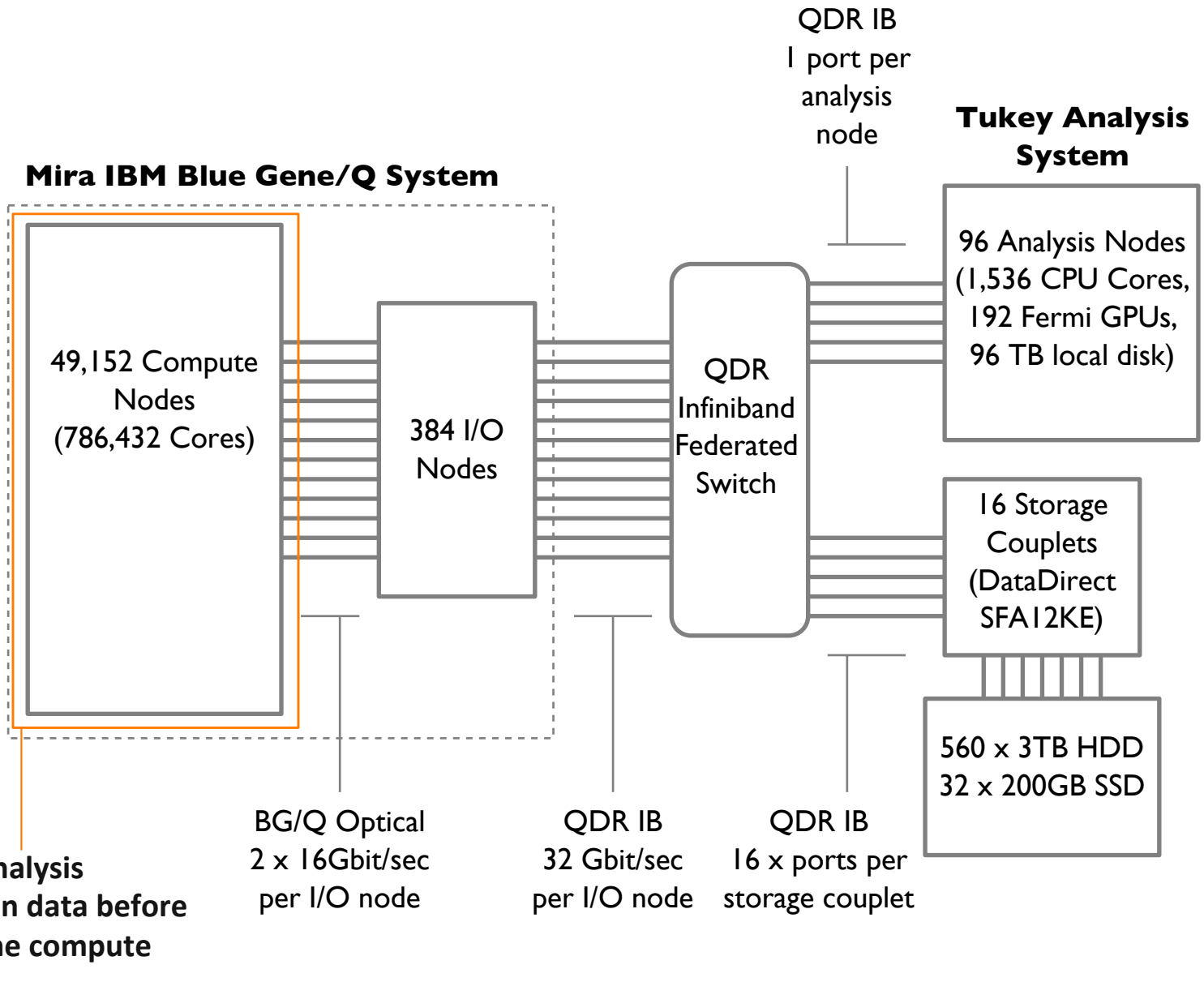
High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

Analyzing Data: Co-Analysis



High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

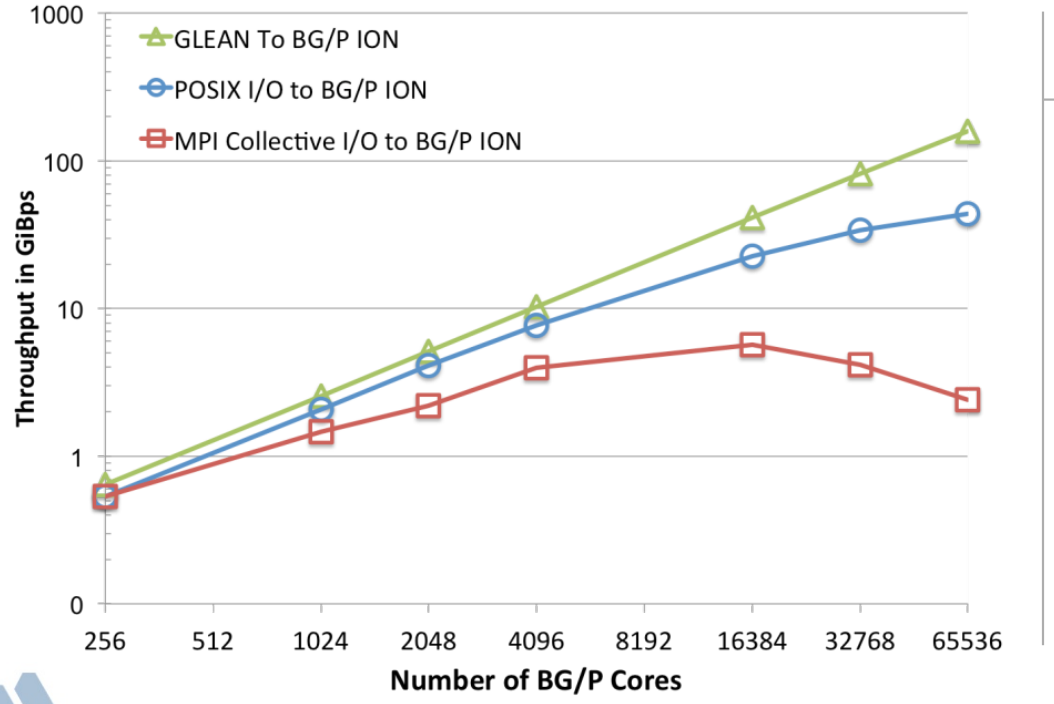
Analyzing Data: In Situ Analysis



High-level diagram of 10 Pflop IBM Blue Gene/Q system at Argonne Leadership Computing Facility

Streamlining Data Movement in Airflow Simulation

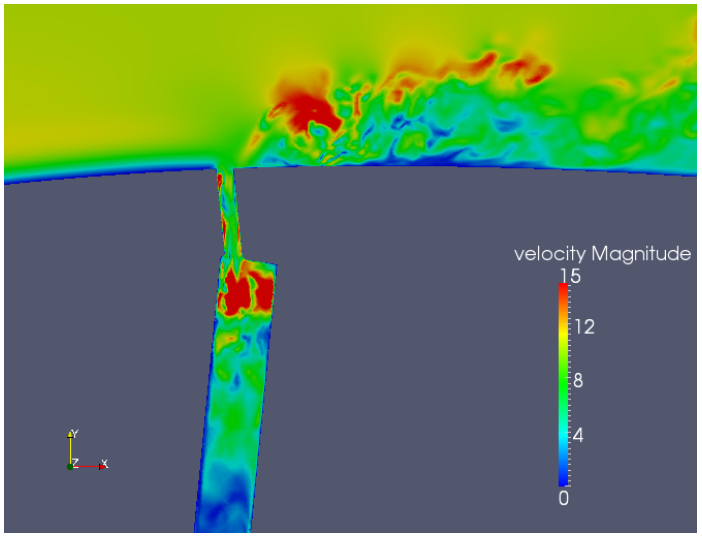
- PHASTA CFD simulations produce as much as ~200 GB per time step
  - Rate of data movement off compute nodes determines how much data the scientists are able to analyze
- GLEAN is a flexible and extensible framework for simulation-time data movement and analysis
  - Accelerating I/O via topology awareness, asynchronous I/O
  - Enabling in situ analysis and co-analysis

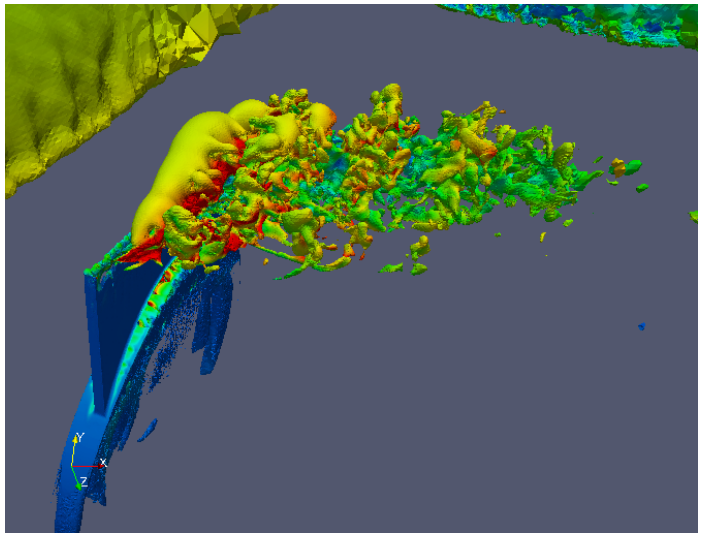


Strong scaling performance for 1GB data movement off ALCF Intrepid Blue Gene/P compute nodes. GLEAN provides 30-fold improvement over POSIX I/O at large scale. Strong scaling is critical as we move towards systems with increased core counts.

Thanks to V. Vishwanath (ANL) for providing this material.

Observing Simulated Synthetic Jet Behavior





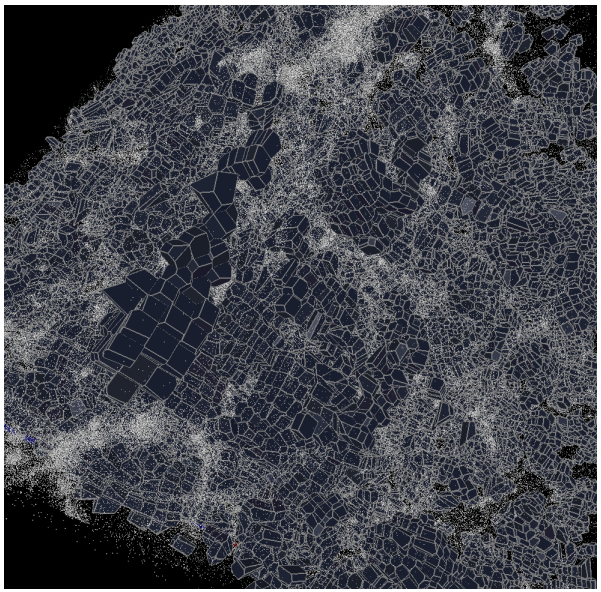
Cut plane through synthetic jet (left) and isosurface of vertical velocity (right) colored by velocity (both for 3.3 billion element mesh). Analysis performed with ParaView.

Thanks to K. Jansen (UC Boulder) for these images.

- Using GLEAN, scientists are able to use co-analysis to observe simulation behavior at run time and avoid storage bottlenecks
  - In co-analysis, data is moved from compute to analysis resources without first being stored on disk
  - Reduces storage requirements, overlaps analysis with simulation, and achieves very data throughput (48 GiBps)
- This enables the scientists to better understand the temporal characteristics of the synthetic jet
  - Cost of analyzing a timestep is much lower, so scientists can view results at a higher temporal fidelity than was feasible before (approx. every 10 timesteps)

Other Work

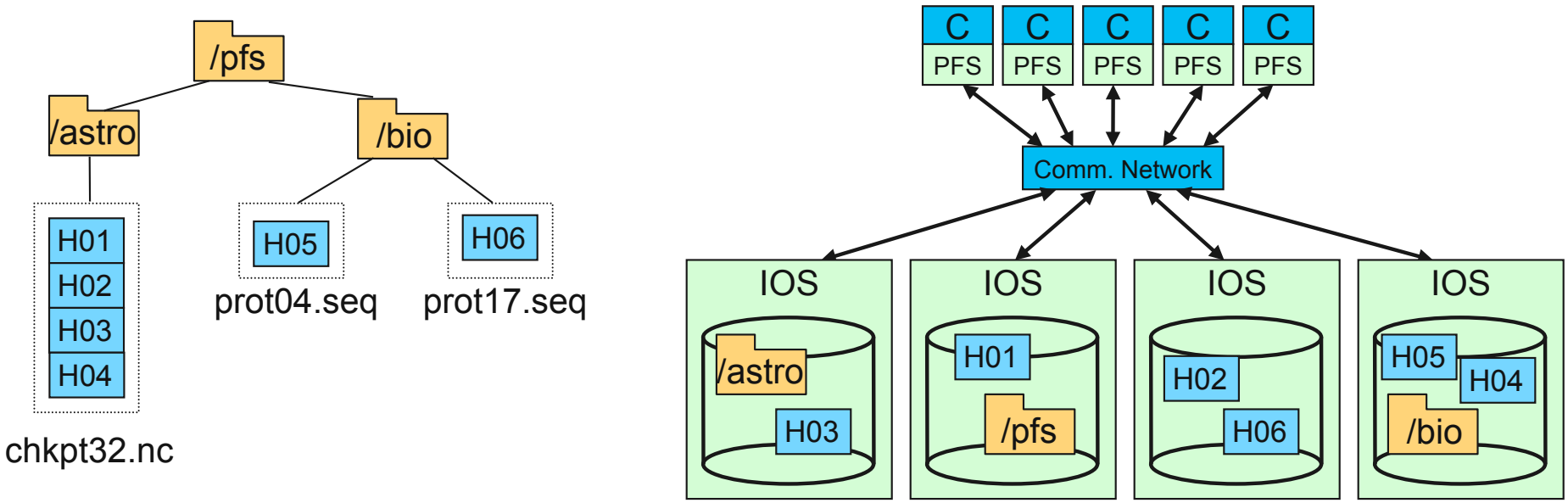
- Damaris
  - See Matthieu's talk later today!
  - M. Dorier et al. Damaris: Leveraging Multicore Parallelism to Mask I/O Jitter. INRIA Technical Report 7706. August, 2011.
- ADIOS
  - Framework for I/O and data analysis in HPC systems
  - http://www.olcf.ornl.gov/center-projects/adios/
- Do It Yourself (DIY)
  - Library of functions for developing analysis functionality, including integrated into application codes
  - T. Peterka et al. Scalable Parallel Building Blocks for Custom Data Analysis. LDAV 2011. October, 2011.



DIY is being used to calculate Voronoi tesselation during runtime in HACC cosmology code.

Object Storage as a File System Alternative

Parallel File Systems: What They Do



An example parallel file system, with large astrophysics checkpoints distributed across multiple I/O servers (IOS) while small bioinformatics files are each stored on a single IOS.

- Distribute data across many servers

  (e.g., by managing large collection of objects)
- Manage associated metadata (e.g., permissions, owners, relationships)
- Provide a POSIX file "veneer" atop distributed data

  (e.g., by mapping a POSIX file abstraction onto a set of objects)

PFSes: Successes and Failure

- Successes
  - Current parallel file system designs scale to hundreds of servers (Big!)
  - Individual servers can move data very effectively, given the right patterns (Fast!)
  - Name space is not loved, but mostly ok unless we are creating files for every process
- Failure
  - The POSIX file model provides a single byte stream into which data must be stored
  - HPC applications create complex output that are naturally multi-stream
    - Structured datasets (e.g., HDF5, netCDF)
    - Log-based datasets (e.g., PLFS, ADIOS BP)
  - Dilemma:
    - Do I create lots of files to hold many streams? Stresses the metadata subsystem!
    - Do I map many streams into a single file? I need to manage distribution and locking!

The Capt. Kirk Solution*

- If software presents a no-win situation, recode it
- Expose individual object byte streams for use by I/O libraries (e.g., Parallel netCDF, PLFS)
  - Library becomes responsible for mapping its data structures into these objects
- Keep the rest!
  - Have directories, organize objects under file names
  - Keep permissions, etc.

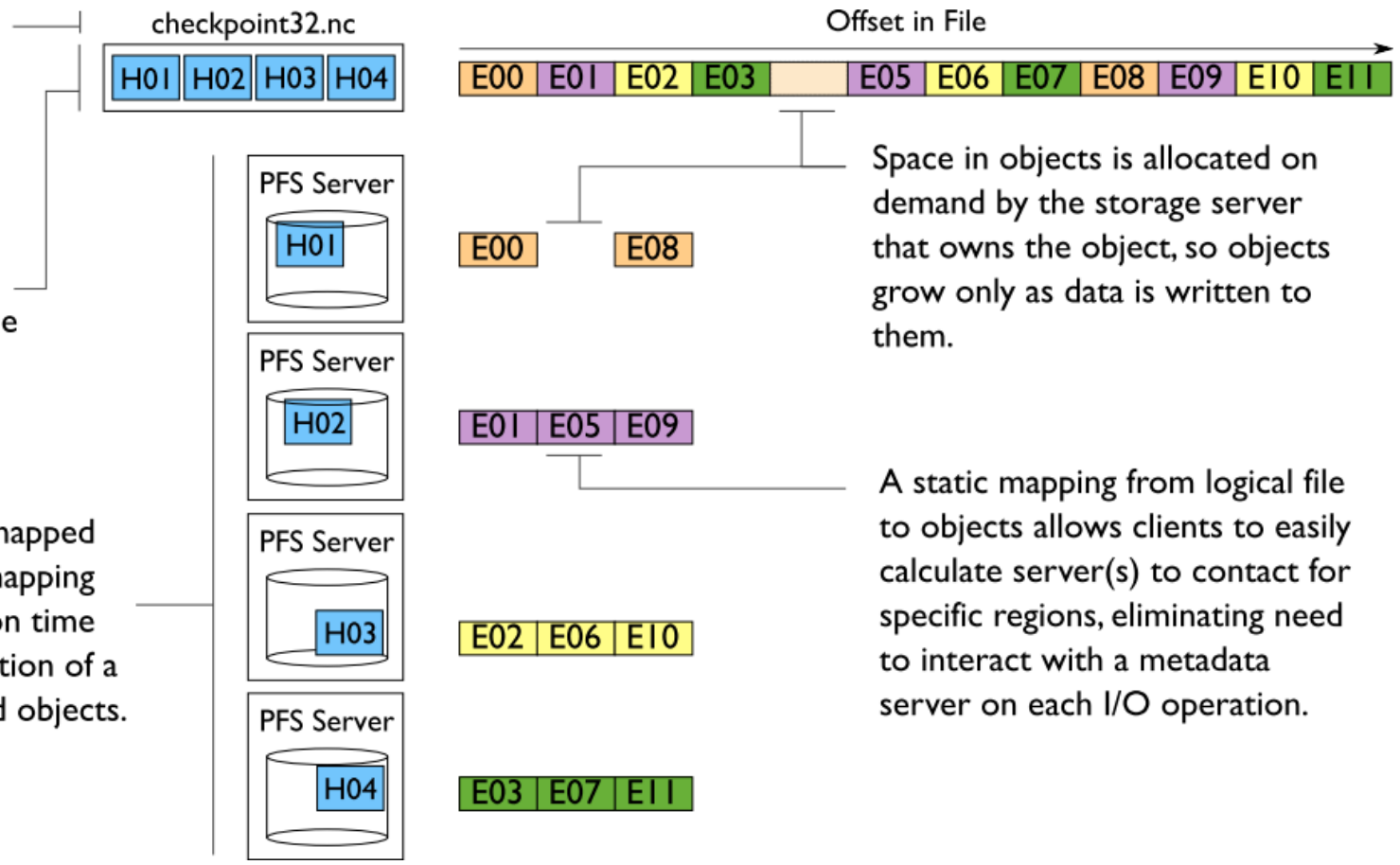
* See http://en.wikipedia.org/wiki/Kobayashi_Maru

How POSIX Veneer is Implemented

Logically a file is an extendable sequence of bytes that can be referenced by offset into the sequence.

Metadata associated with the file specifies a mapping of this sequence of bytes into a set of objects on PFS servers.

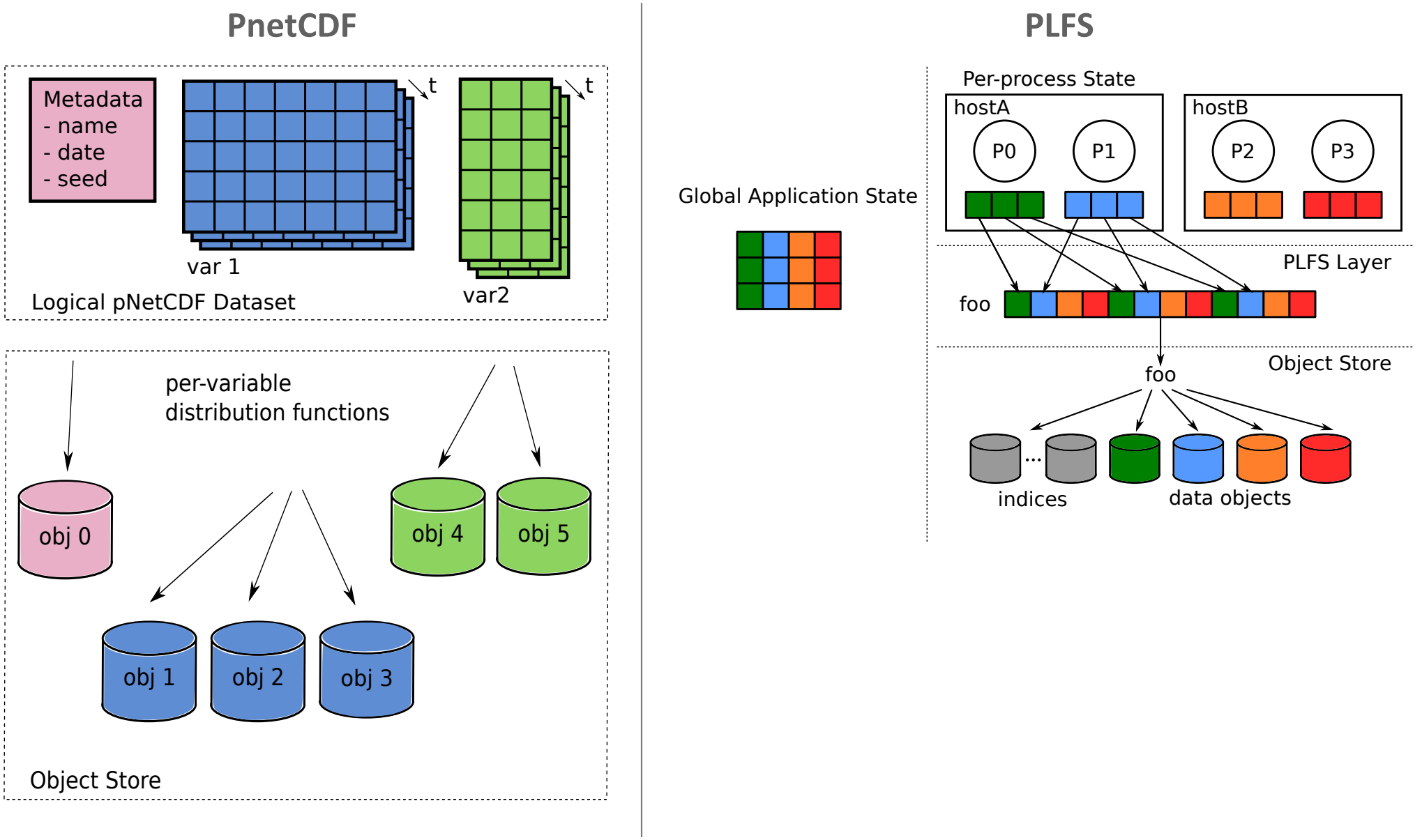
Extents in the byte sequence are mapped into objects on PFS servers. This mapping is usually determined at file creation time and is often a round-robin distribution of a fixed extent size over the allocated objects.



Space in objects is allocated on demand by the storage server that owns the object, so objects grow only as data is written to them.

A static mapping from logical file to objects allows clients to easily calculate server(s) to contact for specific regions, eliminating need to interact with a metadata server on each I/O operation.

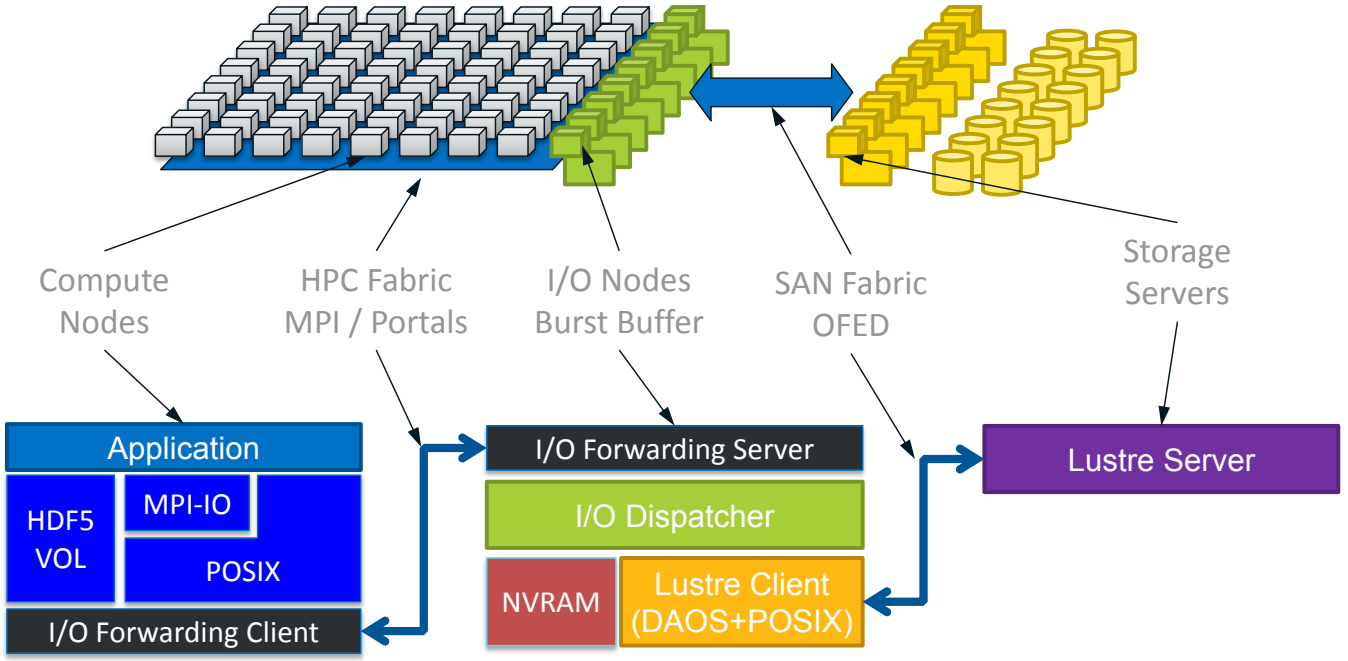
Mapping Abstractions to Objects: PnetCDF and PLFS



S. Kim et al. An Evolutionary Path to Object Storage Access. PDSW 2012. November, 2012.

Other Work

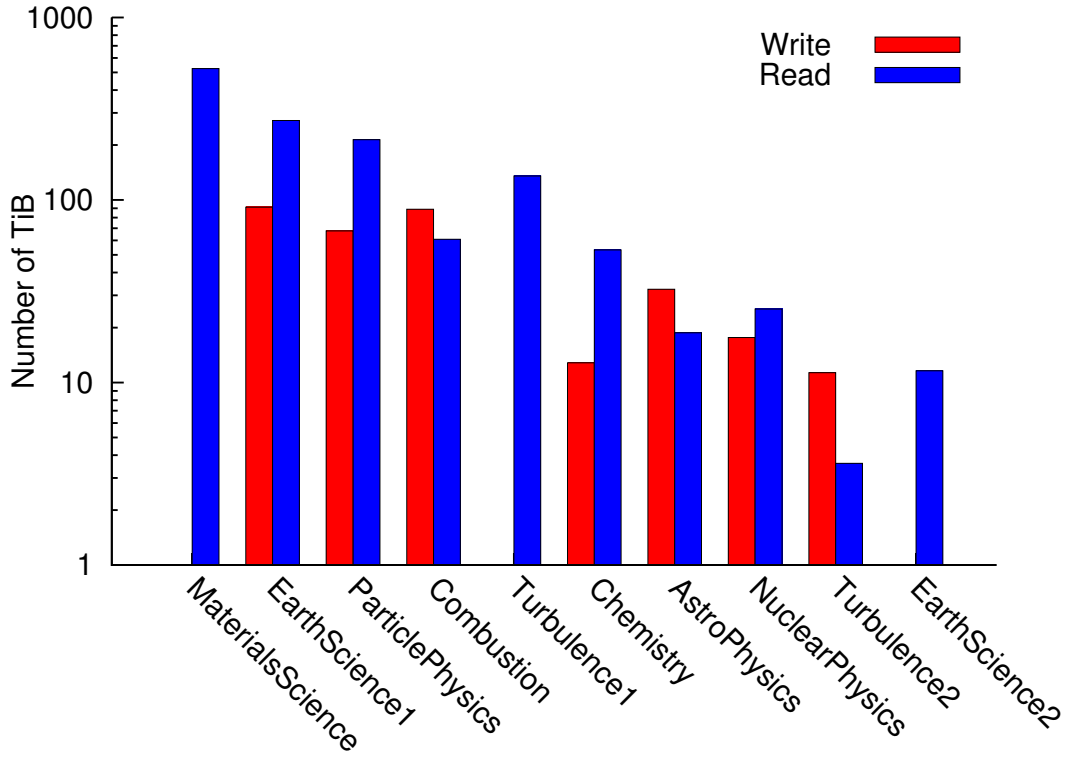
- DataMods
  - Includes notion of defining access methods
  - N. Watkins et al. DataMods: Programmable File System Services. PDSW 2012. November, 2012.
- DOE Fast Forward
  - End-to-end push on future software stack, possibly including device mgmt.
  - E. Barton et al. Fast Forward I/O and Storage. PDSW 2012. November, 2012. (Diagram from this presentation)



Big Data and Extreme Scale I/O

Big Data on Leadership Platforms

"Very few large scale applications of practical importance are NOT data intensive." – Alok Choudhary, IESP, Kobe, Japan, April 2012



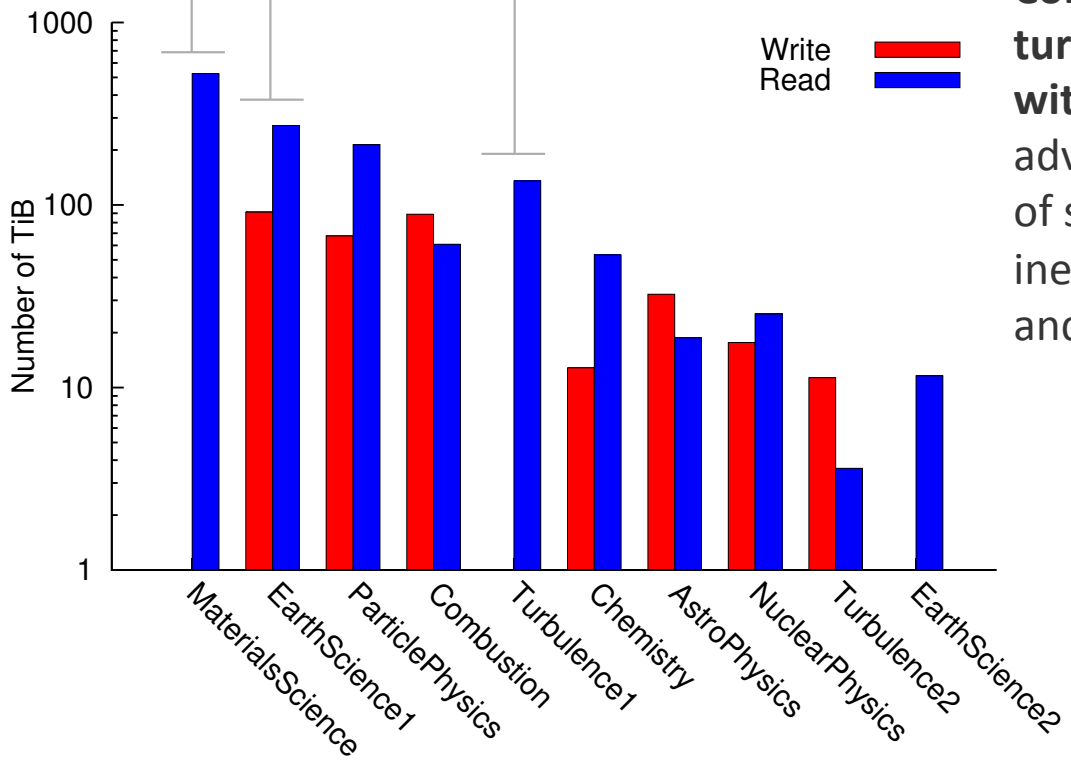
Top 10 data producer/consumers instrumented with Darshan over the month of July, 2011 on Intrepid BG/P system at Argonne. Surprisingly, three of the top producer/consumers almost exclusively read existing data.

Big Data on Leadership Platforms

Matching large scale simulations of dense suspensions with empirical measurements to better understand properties of complex materials such as concrete.

Processing large-scale seismographic datasets to develop a 3D velocity model used in developing earthquake hazard maps.

Comparing simulations of turbulent mixing of fluids with experimental data to advance our understanding of supernovae explosions, inertial confiment fusion, and supersonic combustion.



Top 10 data producer/consumers instrumented with Darshan over the month of July, 2011 on Intrepid BG/P system at Argonne. Surprisingly, three of the top producer/consumers almost exclusively read existing data.

Lab-Wide Data Service

Advanced Photon Source

External sources

Data flows in TB/day

163

9

9

75

100

Short-term storage

10

50

Long-term storage

100

150

Other sources that remain to be quantified

Diagram thanks to I. Foster.

Wrapping Up

Time of Change

- In-system storage
  - Opening up new opportunities for checkpoint, analysis, staging
  - Raising questions of ownership, modifications to system software
- Alternatives to traditional post-processing analysis
  - Working around portions of the I/O bottleneck
  - Raising questions about how to engineer analysis at run-time
- New storage system models
  - Providing alternatives to the POSIX model
  - Raising questions about what to present to users, how to evolve
- Big Data
  - Creating new demands on I/O systems, new workloads
  - Raising questions of the role of storage systems not only for HPC, but as a laboratory resource

Acknowledgments

This work was supported by the Director, Office of Advanced Scientific Computing Research, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-06CH11357, including through the Scientific Discovery through Advanced Computing (SciDAC) Institute of Scalable Data Management, Analysis and Visualization.