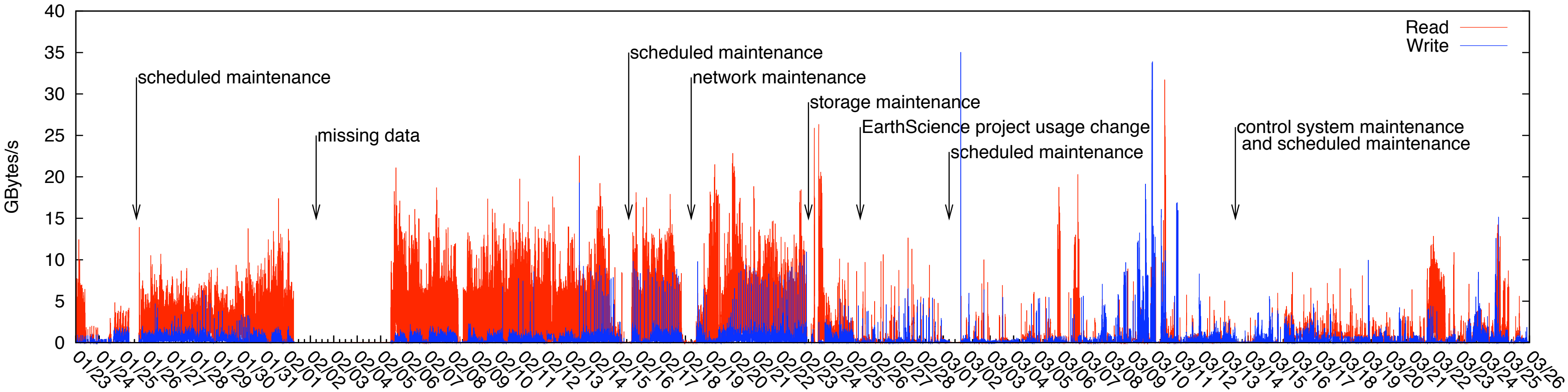
STUDYING I/O IN THE DATA CENTER:
OBSERVING AND SIMULATING I/O FOR FUN AND PROFIT

ROB ROSS
Mathematics and Computer Science Division
Argonne National Laboratory
rross@mcs.anl.gov

June 23, 2016

I/O: A STORAGE DEVICE PERSPECTIVE (2011)



Aggregate I/O throughput on BG/P storage servers at one minute intervals.

- The I/O system is rarely **idle** at this granularity.
- The I/O system is also rarely at more than 33% of peak bandwidth.

- What's going on?
  - Are applications not really using the FS?
  - Is it just not possible to saturate the FS?
  - Did we buy too much storage?

P. Carns et al. "Understanding and improving computational science storage access through continuous characterization." ACM Transactions on Storage. 2011.

Argonne NATIONAL LABORATORY

OBSERVATION: DARSHAN

WHAT ARE APPLICATIONS DOING, EXACTLY?

Work was inspired by the CHARISMA project (mid 1990s), which had the goal of capturing the behavior of several production workloads, on different machines, at the level of individual reads and writes.

- Studied Intel iPSC/860 and Thinking Machines CM-5
- Instrumented FS library calls
- Complete tracing of thousands of jobs in total
  - Jobs: sizes, numbers of concurrent jobs
  - File access patterns: local vs. global, access size, strided access

- The community has continued to develop new tracing and profiling techniques and apply them to critical case studies
- … but tracing at a broad, system-wide level is out of the question on modern systems at this point

Nieuwejaar, Nils, et al. "File-access characteristics of parallel scientific workloads." IEEE Transactions on Parallel and Distributed Systems. 7.10 (1996): 1075-1089.

DARSHAN: CONCEPT

Goal: to observe I/O patterns of the majority of applications running on production HPC platforms, without perturbing their execution, with enough detail to gain insight and aid in performance debugging.

- Majority of applications – integrate with system build environment
- Without perturbation – bounded use of resources (memory, network, storage); no communication or I/O prior to job termination; compression.
- Adequate detail:
  - basic job statistics
  - file access information from multiple APIs

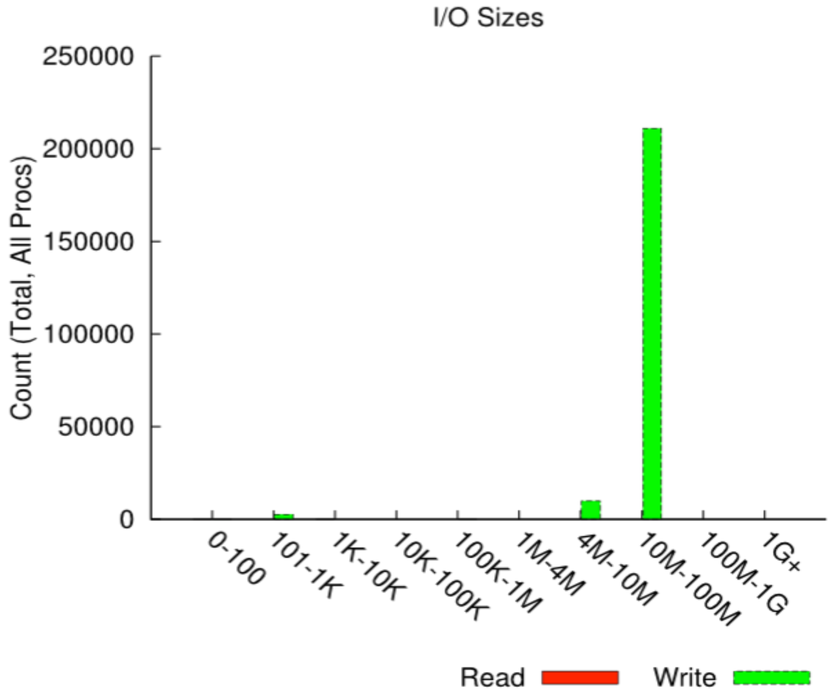
DARSHAN: TECHNICAL DETAILS

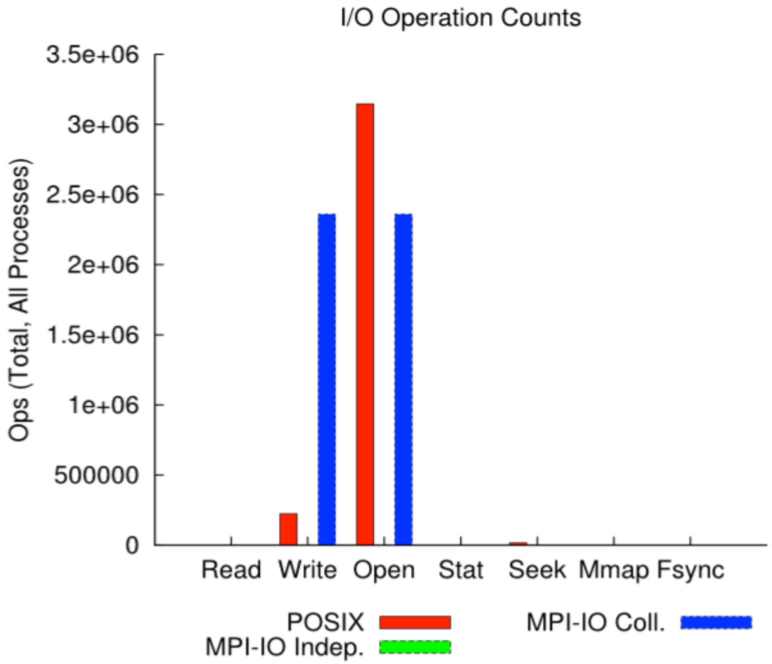
- Runtime library for characterization of application I/O
  - Transparent instrumentation
  - Captures POSIX I/O, MPI-IO, and limited HDF5 and PNetCDF functions
- Minimal application impact ("just leave it on")
  - Bounded memory consumption per process
  - Records strategically chosen counters, timestamps, and histograms
  - Reduces, compresses, and aggregates data at MPI_Finalize() time
  - Generates a single compact summary log per job
- Compatible with IBM BG, Cray, and Linux environments
  - Deployed system-wide or enabled by individual users
  - No source code modifications or changes to build rules
  - No file system dependencies
- Enabled by default at NERSC, ALCF, and NCSA
- Also deployed at OLCF, LLNL, and LANL in limited use

http://www.mcs.anl.gov/research/projects/darshan

DARSHAN: I/O CHARACTERIZATION

786,432 process turbulence simulation on Mira





File Count Summary
(estimated by I/O access offsets)

type	number of files	avg. size	max size
total opened	17	199G	1.6T
read-only files	1	2.0K	2.0K
write-only files	13	260G	1.6T
read/write files	0	0	0
created files	13	260G	1.6T

Most Common Access Sizes

access size	count
16777216	210977
8388608	9866
256	2598
68	9

SHARING DATA WITH THE COMMUNITY

This type of data can be invaluable to researchers trying to understand how applications use HPC systems.

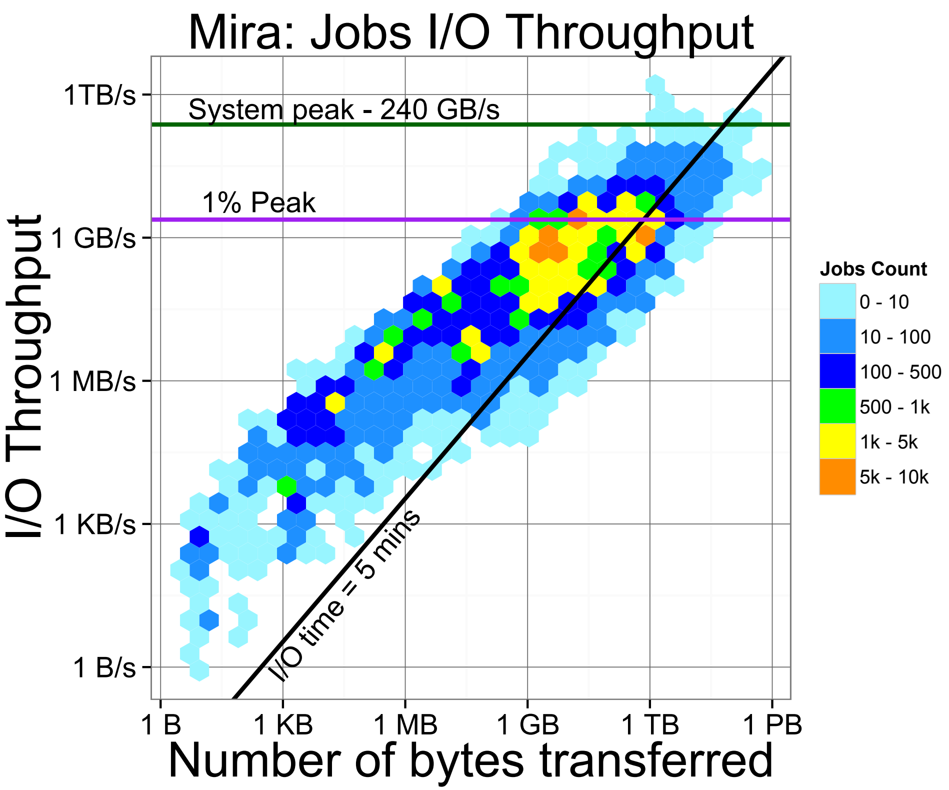
- Conversion utilities can anonymize and re-compress data
- Compact data format in conjunction with anonymization makes it possible to share data with the community in bulk
- ALCF I/O Data Repository provides access to production logs captured on Intrepid
- Logs from 2010 to 2013, when the machine was retired

ALCF I/O Data Repository Statistics	
Unique log files	152,167
Core-hours instrumented	721 million
Data read	25.2 petabytes
Data written	5.7 petabytes

http://www.mcs.anl.gov/research/projects/darshan/data/

DARSHAN: BUILDING HIGHER-LEVEL VIEWS

- Mining Darshan data
  - Ongoing work led by M. Winslett (UIUC)
  - Mining Darshan logs for interactive visualization and generation of representative workloads



Web dashboard for ad-hoc analysis: https://github.com/huongluu/DarshanVis

Luu, Huong, et al. "A multiplatform study of I/O behavior on petascale supercomputers." *Proceedings of HPDC*, 2015.

TARGETING ATTENTION: SMALL WRITES TO SHARED FILES

- Small writes can contribute to poor performance as a result of poor file system stripe alignment, but there are many factors to consider:
  - Writes to non-shared files may be cached aggressively
  - Collective writes are normally optimized by MPI-IO
  - Throughput can be influenced by additional factors beyond write size
- We searched for jobs that wrote less than 1 MiB per operation to shared files without using any collective operations
- Candidates for collective I/O or batching/buffering of write operations

Summary of matching jobs:

Thresholds	> 100 million small writes 0 collective writes
Total jobs analyzed	261,890
Jobs matching heuristic	220
Unique users matching heuristic	11
Largest single-job small write count	5.7 billion

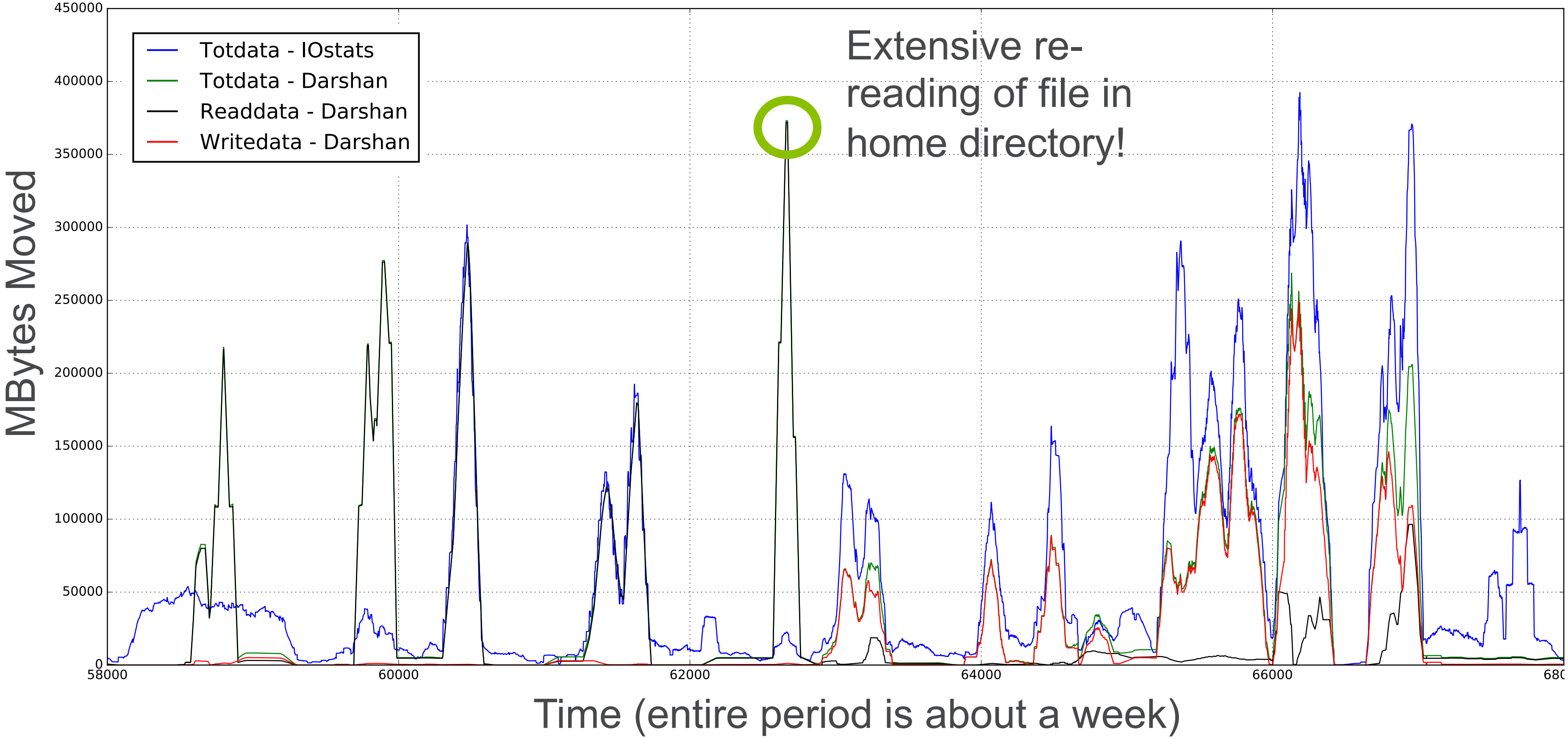
<u>Top example</u>

- Scale: 128 processes
- Run time: 30 minutes
- Max. I/O time per process: 12 minutes
- Metric: issued 5.7 billion write operations, each less than 100 bytes in size, to shared files
- Averaged just over 1 MiB/s per process during shared write phase

P. Carns et al, “Production I/O Characterization on the Cray XE6,” CUG 2013.

Argonne NATIONAL LABORATORY

FIRST STEPS IN APPLYING ML TO DARSHAN



Data from Sandeep Madireddy (ANL).

DARSHAN: STATUS

http://www.mcs.anl.gov/research/projects/darshan/

- Open source (BSD-like license)
- Git repository: https://xgitlab.cels.anl.gov/groups/darshan
  - Source code access
  - Issue tracking
- Users mailing list
- Routinely used on IBM BG/Q, Cray XC30 and XC40, and Linux
- Enabled by default on Mira, Cori, Edison, and Blue Waters
- Deep instrumentation:
  - POSIX
  - MPI-IO
  - BG/Q job information
- Shallow instrumentation:
  - Parallel netCDF
  - HDF5

SIMULATION: CODES

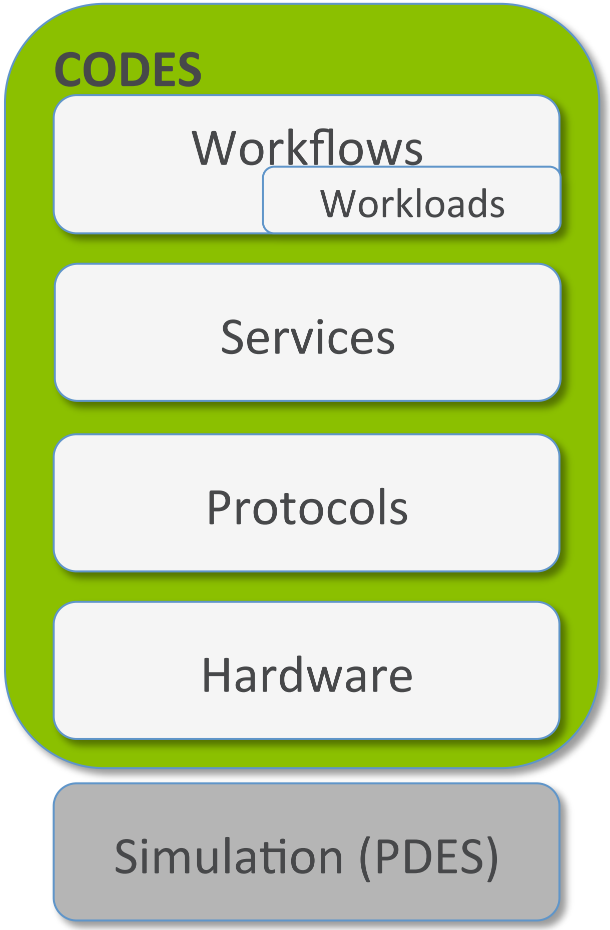
Rensselaer Polytechnic Institute
- Chris Carothers
- Elsa Gonsiorowski
- Justin LaPre
- Caitlin Ross
- Noah Wolfe

Argonne National Laboratory
- Philip Carns
- John Jenkins
- Misbah Mubarak
- Shane Snyder
- Rob Ross

CODES

The goal of CODES is to use highly parallel simulation to explore exascale and distributed data-intensive storage system design.

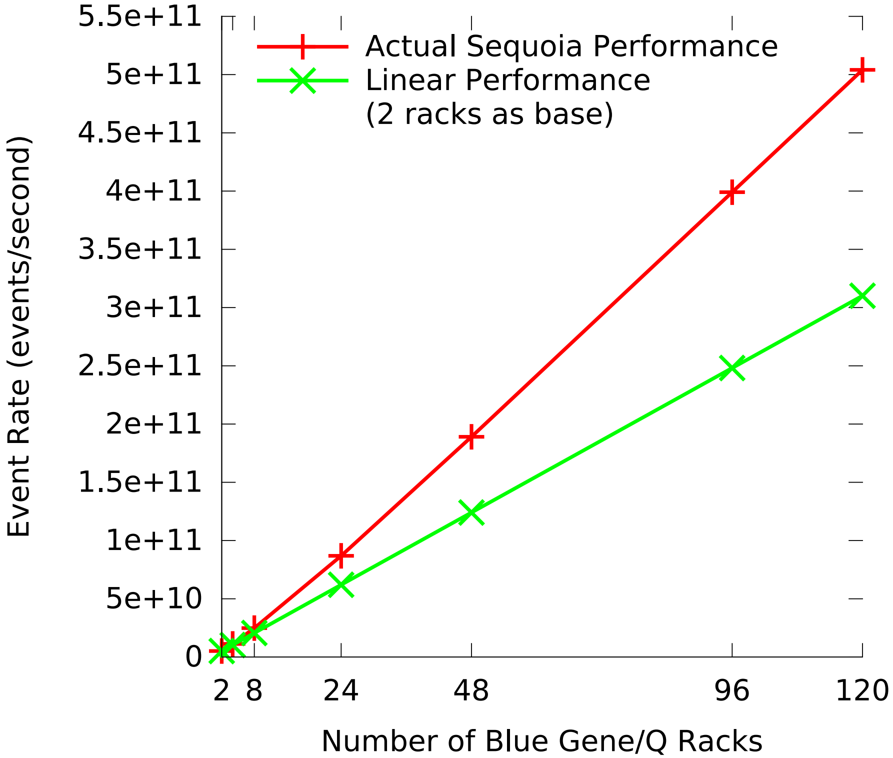
- Project kickoff in 2010
- Set of models, tools, and utilities intended to simplify complex model specification and development
  - Configuration utilities
  - Commonly-used models (e.g., networking)
  - Facilities to inject application workloads
- Using these components to understand systems of interest to DOE ASCR



PARALLEL DISCRETE EVENT SIMULATION

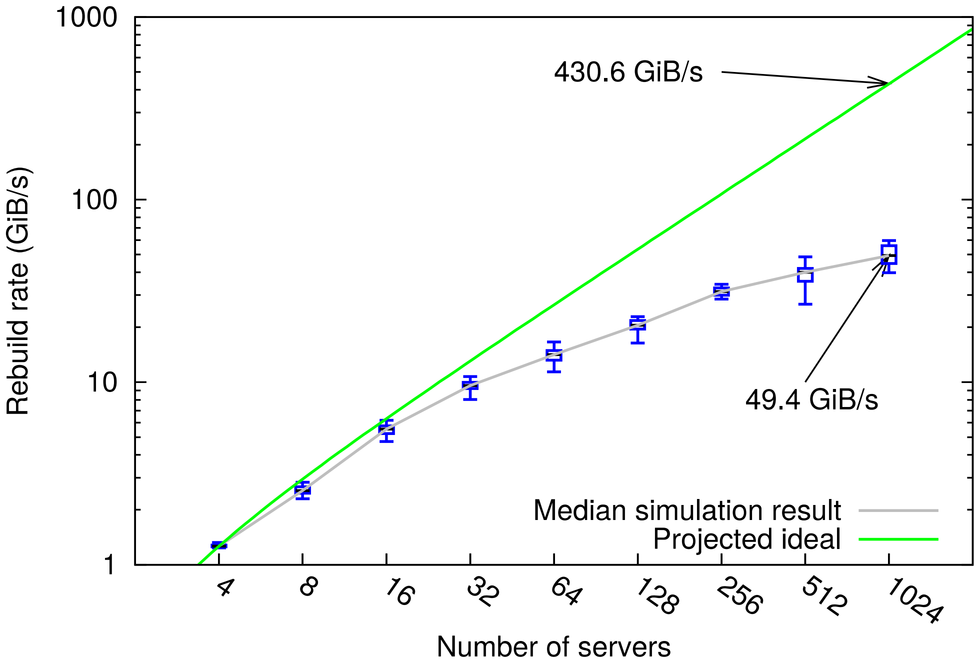
The underpinnings of CODES

- Set of discrete components that send time stamped messages to each other.
- Matches well with systems we want to simulate – workloads contain discrete actions, protocols have discrete, well-defined steps, etc.
- We use **ROSS** (Rensselaer Optimistic Simulation System)
  - Builds on MPI for parallelism
  - Utilizes Time Warp/Optimistic or YAWNs/Conservative schedulers
  - Users provide functions for *reverse computation* – undo the effects of a particular event on the LP state



Barnes et. al. "Warp speed: Executing time warp on 1,966,080 cores," in Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (SIGSIM-PADS'13)

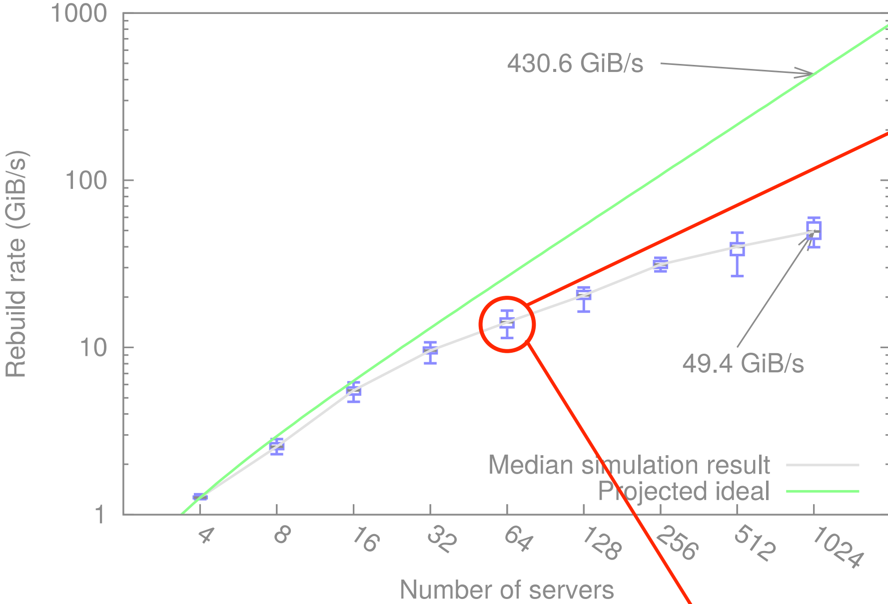
CODES: UNDERSTANDING OBJECT STORES

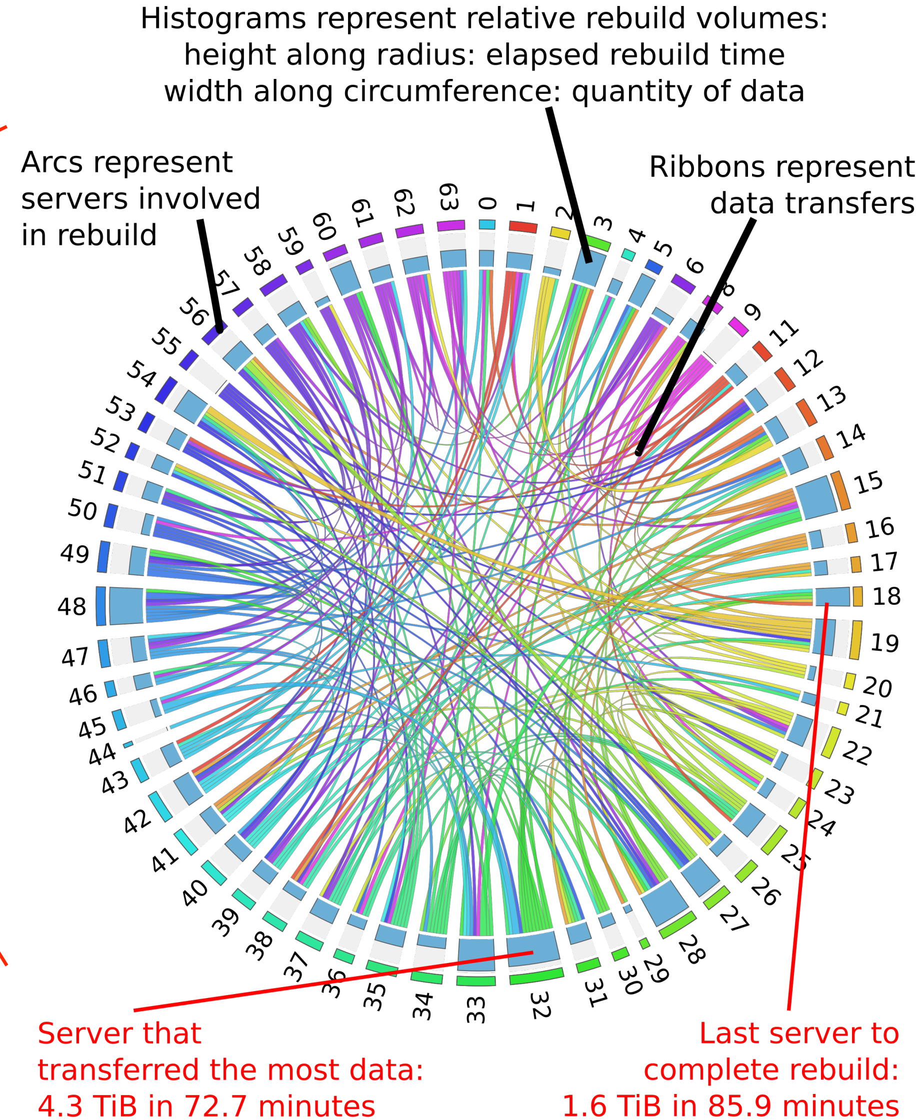


P. Carns et al. “The Impact of Data Placement on Resilience in Large-Scale Object Storage Systems.” *MSST.* 2016.

- Investigating replicated object storage system response to a server fault
  - Relevant to declustered RAID discussions in CORAL, etc.
- Simulating rebuild traffic:
  - CRUSH object placement algorithm
  - Standard Ceph configuration
  - Simulating pipelining of data movement between nodes and storage device
  - Similar study could be performed for parity recalculation
- Rebuild rate does not scale well with addition of servers

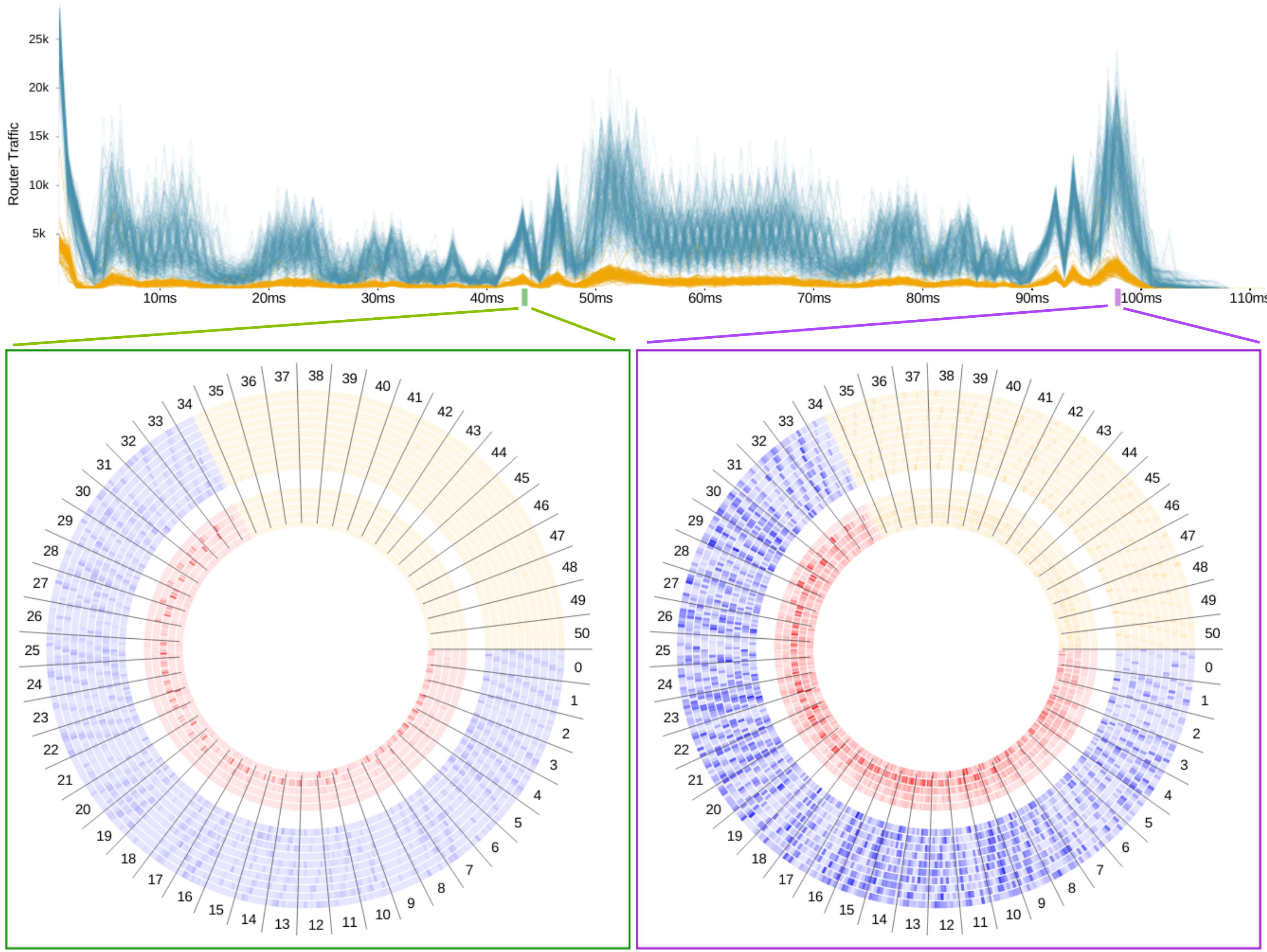
CODES: UNDERSTANDING OBJECT STORES





- Last server to complete the rebuild only interacts with a few peers, one of which is heavily loaded.
- Placement groups aren't sufficiently balancing the load.

CODES: DRAGONFLY NETWORKS



Visualization from K. Li (UC Davis) of 1,728 process AMG comm. pattern with adaptive routing.

CODES: STATUS

http://www.mcs.anl.gov/research/projects/codes/

- Open source (BSD-like license)
- Git repository: https://xgitlab.cels.anl.gov/groups/codes
  - Source code access
  - Issue tracking
- Users mailing list
- Routinely used on IBM BG/P, IBM BG/Q, and Linux
- Repository models include:
  - Torus (configurable dimensionality, etc.)
  - Dragonfly
  - DUMPI trace driver
  - File I/O driver
- Research models:
  - Slim Fly
  - Fat Tree

"Summer of CODES" Workshop
July 12-13, Argonne
Users, Progress, Hackathon
http://press3.mcs.anl.gov/summerofcodes2016/

INTEGRATING DATA: TOKIO

Lawrence Berkeley National Lab

Nick Wright (Lead PI)
Suren Byna
Jialin Liu
Glenn Lockwood
Prabhat
William Woo

Argonne National Laboratory

Philip Carns
Rob Ross
Shane Snyder

Slides in this section were created by Glenn Lockwood.

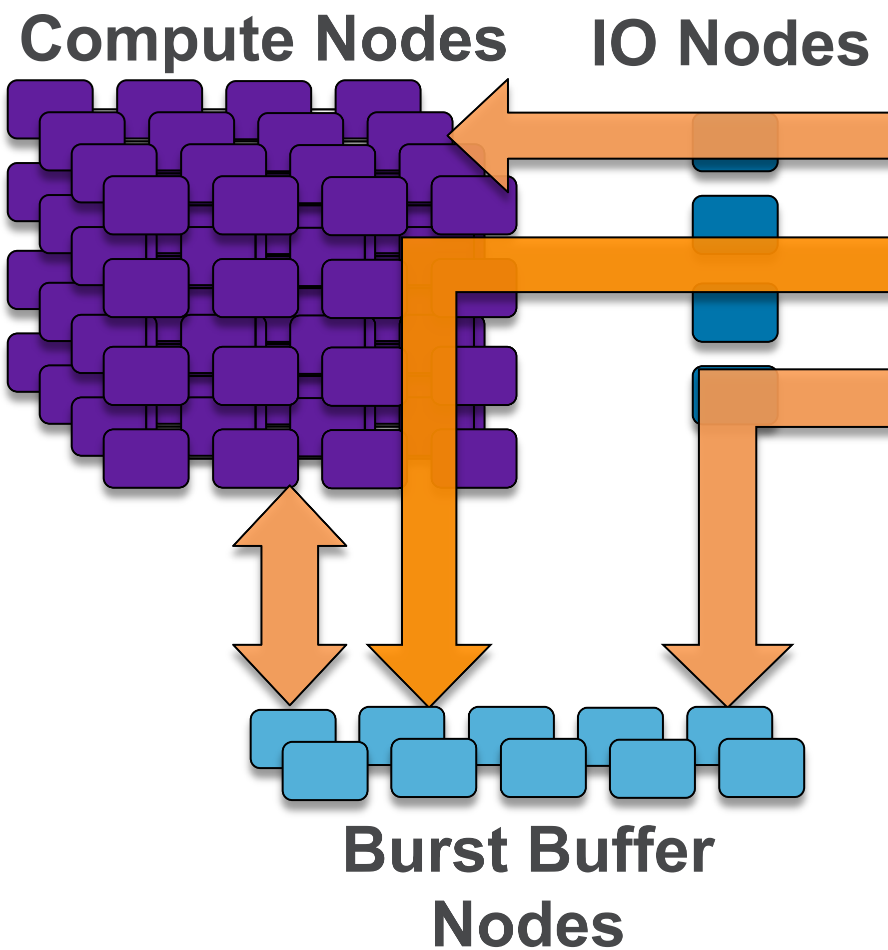
THE TOKIO PROJECT

A holistic view is essential to understanding I/O performance.

Project Goals:

- Development of algorithms and a software framework that will collect and correlate I/O workload data from production HPC resources
- Enable a clearer view of system behavior and the causes of behavior
- Audience: application scientists, facility operators and computer science researchers in the field.

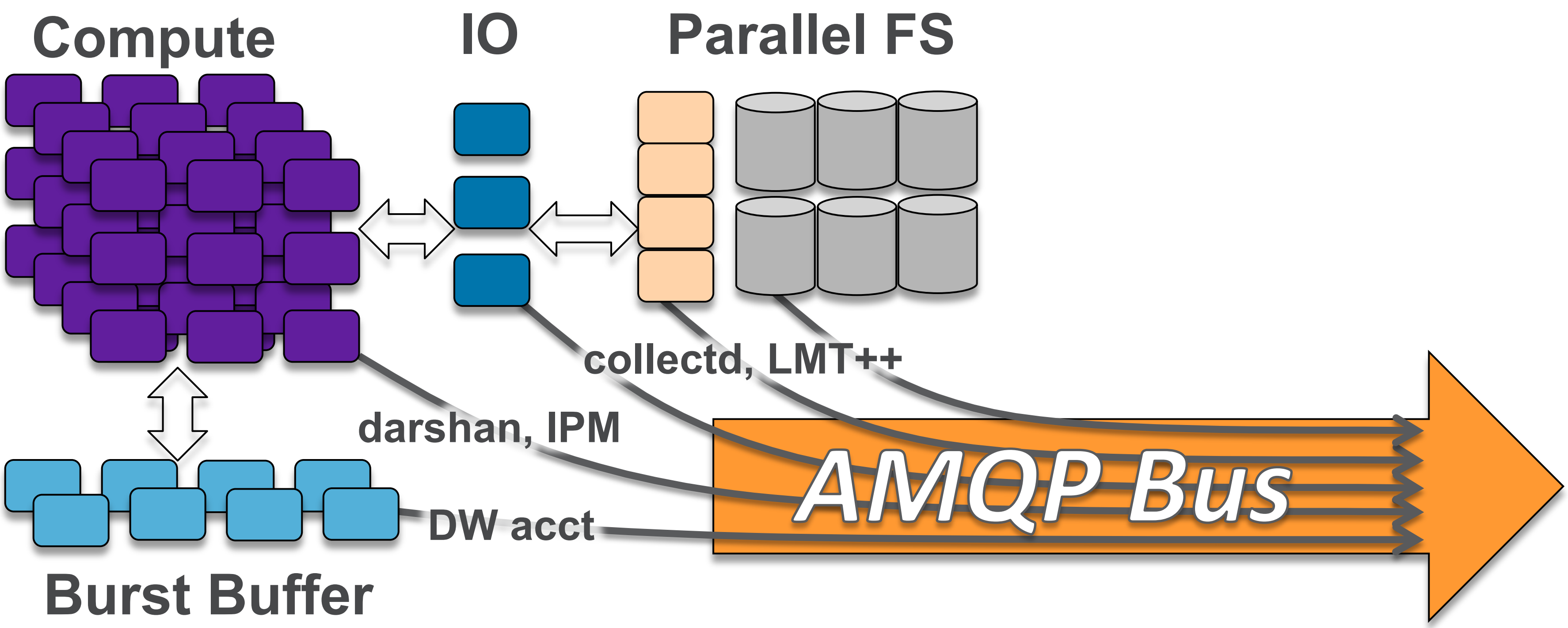
MULTIPLE DATA PATHS IN HIERARCHICAL STORAGE



Burst buffer tier

- data flows through or around flash tier
- one data path between compute and storage is now up to four paths

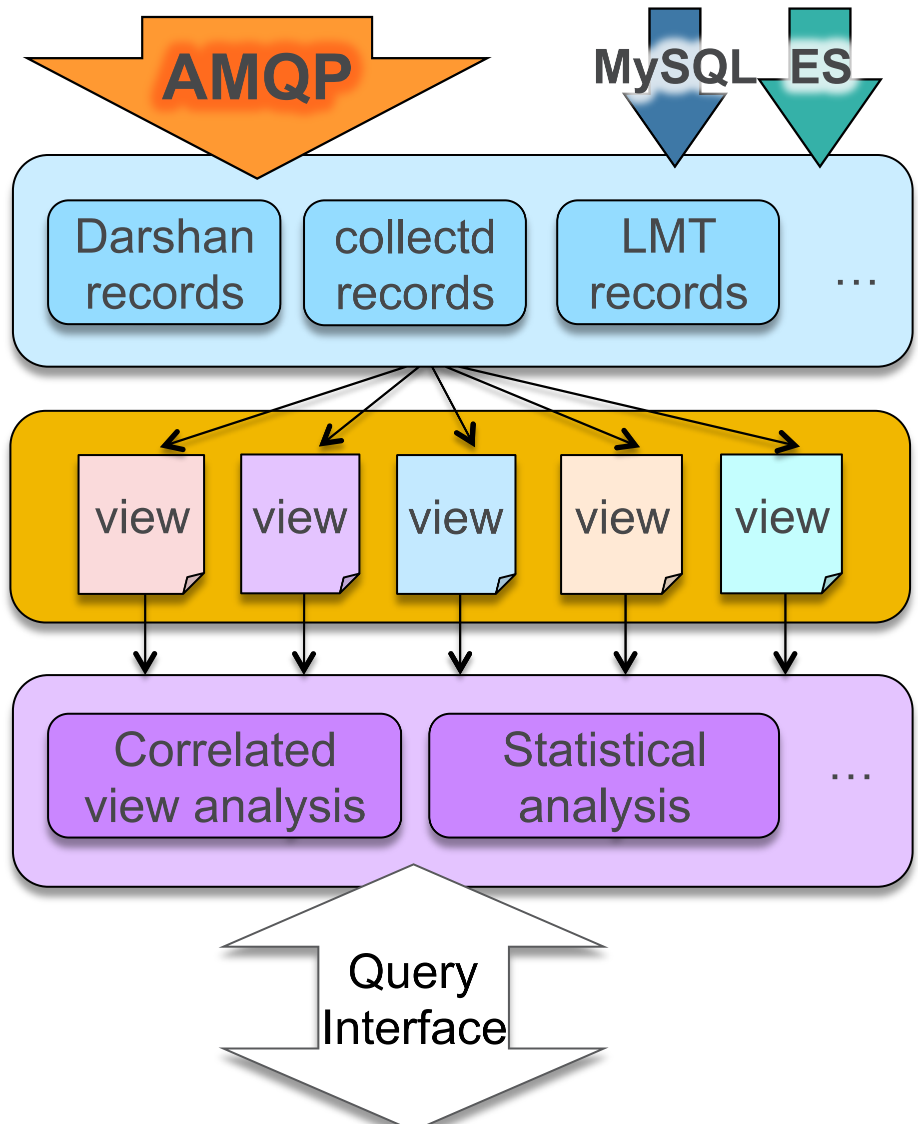
TOKIO FRAMEWORK: SCALABLE COLLECTION



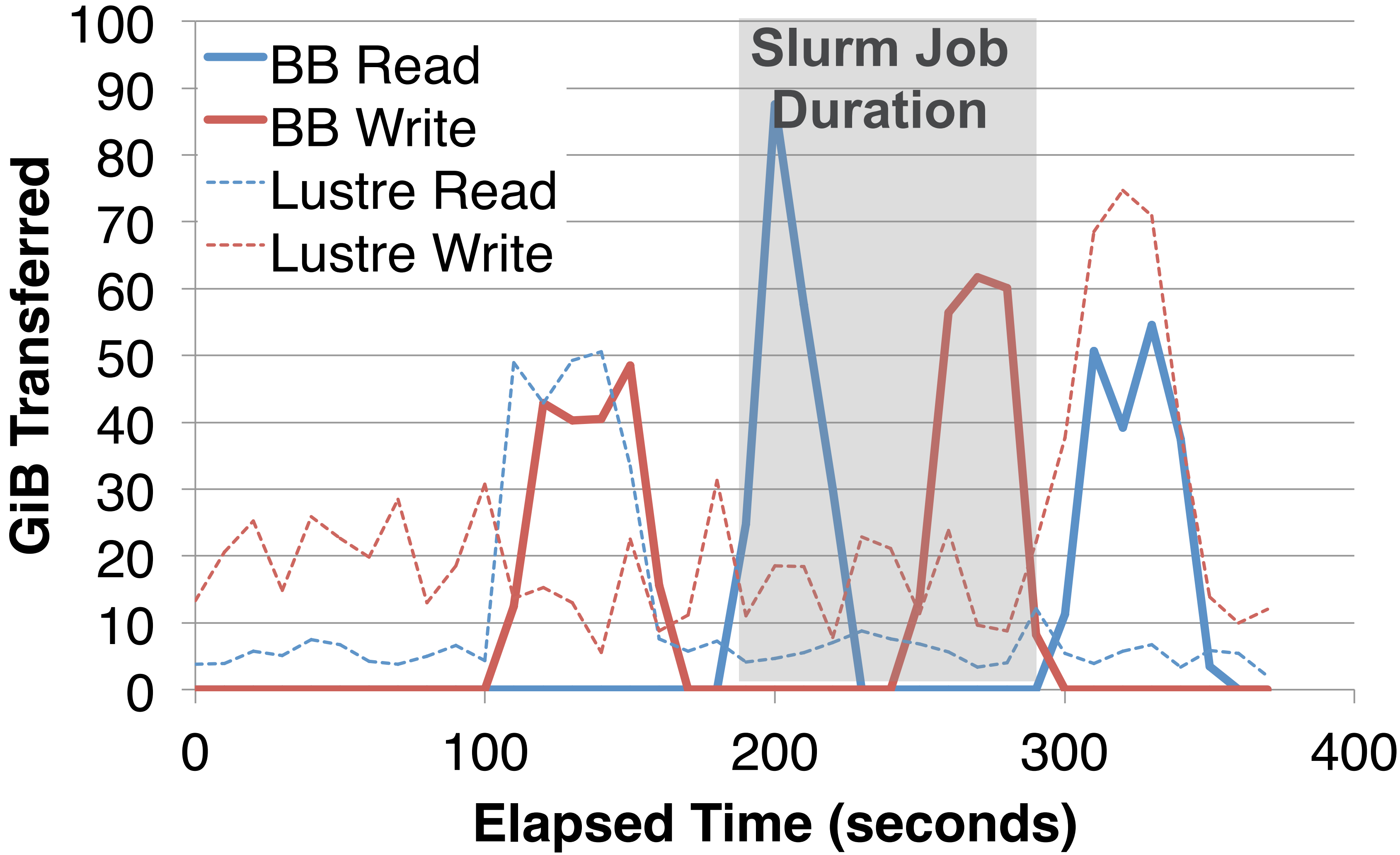
1. **Component-level monitoring data fed into RabbitMQ**
   – Slurm plugins (kernel counters, Darshan, IPM)
   – Native support (procmon, collectd)

TOKIO FRAMEWORK: ANALYTICS FOUNDATION

2. **Component data stored on disk**
   – Native log format or HDF5
   – Divorce collection from parsing
3. **Views: index of on-disk component data**
   – Align data on time, jobid, topology
   – Expose via RDBMS/doc store
4. **Query interface: expose common analytics modules**
   – Web UI, REST API for users
   – Leverage Spark ecosystem

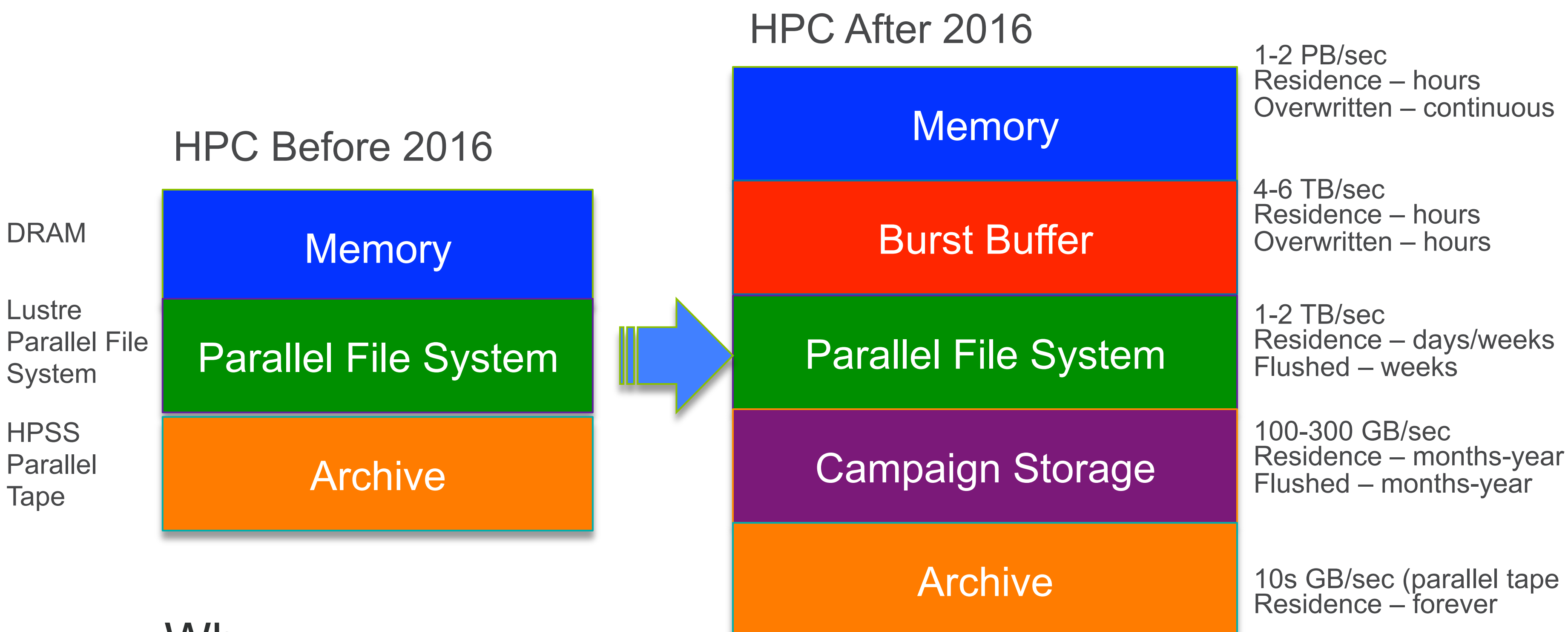


TOKIO: ALIGNING LUSTRE & BURST BUFFER SERVER DATA



WHAT’S NEXT?

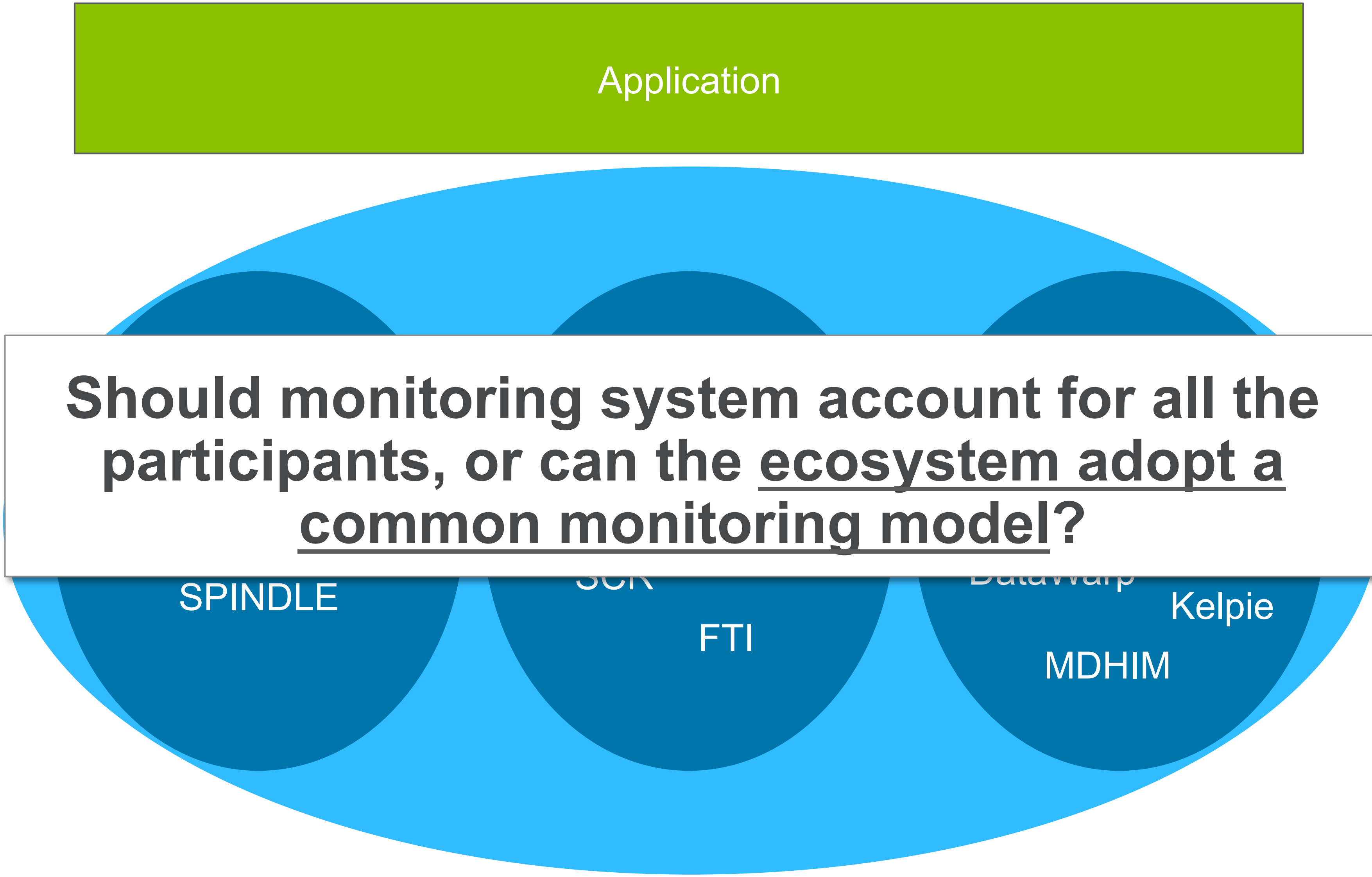
MORE STORAGE LAYERS!



- Why
  - BB: Economics (disk BW/IOPS too expensive)
  - PFS: Maturity and BB capacity too small
  - Campaign: Economics (tape BW too expensive)
  - Archive: Maturity and we really do need a “forever”

Slide from Gary Grider (LANL).

ECOSYSTEM OF DATA SERVICES



Should monitoring system account for all the participants, or can the ecosystem adopt a common monitoring model?

Argonne NATIONAL LABORATORY

THIS WORK IS 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.

Argonne NATIONAL LABORATORY