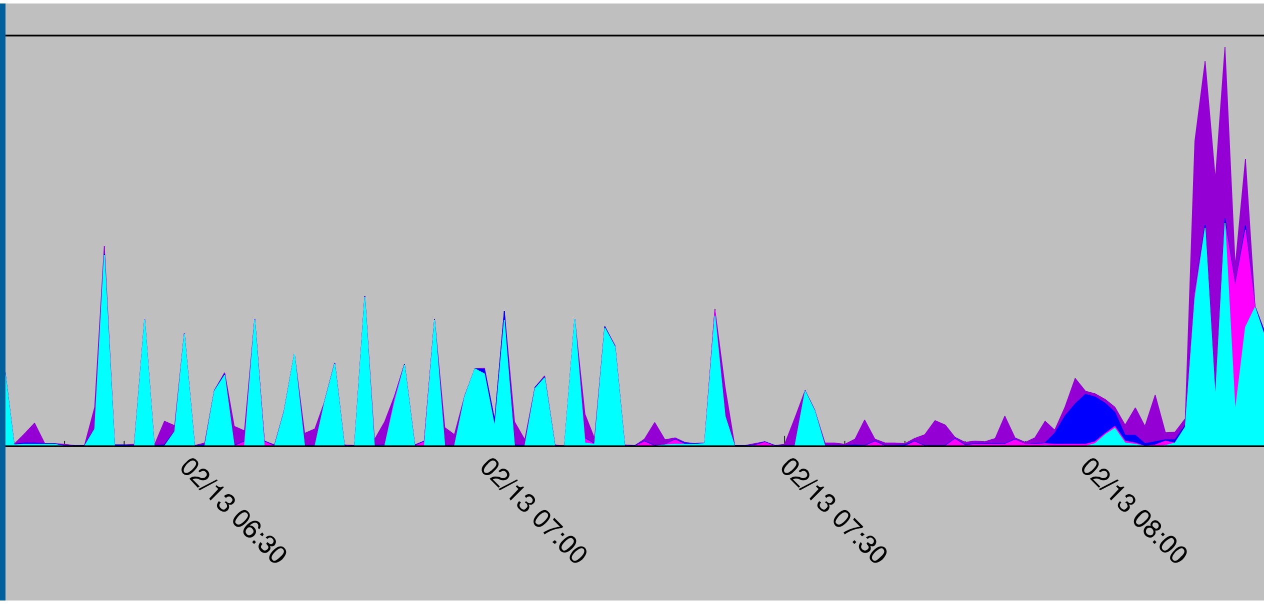
CHARACTERIZING HPC I/O: FROM APPLICATIONS TO SYSTEMS



PHIL CARNS
carns@mcs.anl.gov
Mathematics and Computer Science Division
Argonne National Laboratory

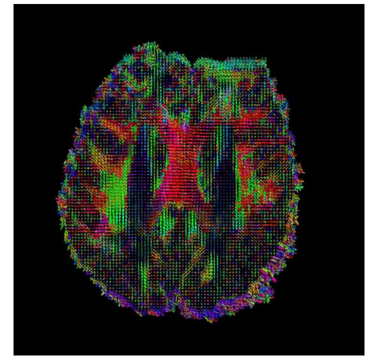
April 20, 2017
TU Dresden

MOTIVATION FOR CHARACTERIZING PARALLEL I/O

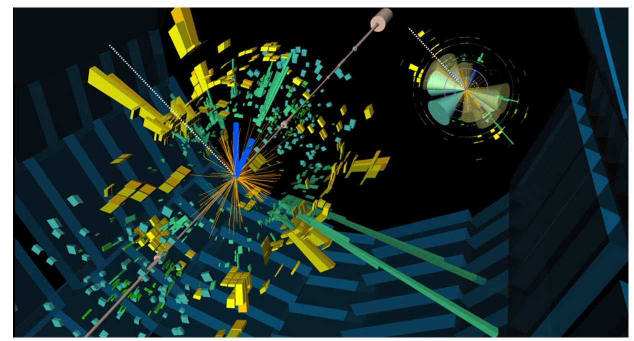
Times are changing in HPC storage!

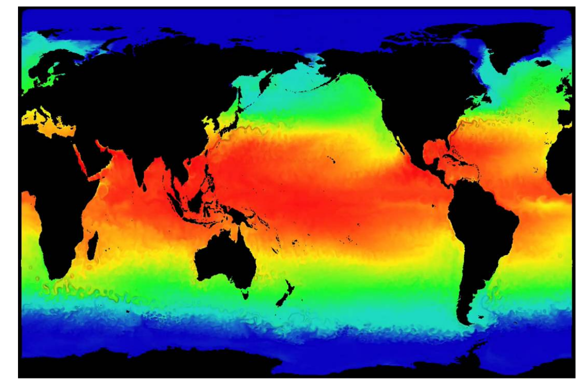
- Most scientific domains are increasingly data intensive: climate, physics, biology and much more
- Upcoming computing platforms include complex, hierarchical storage systems

How can we maximize productivity in this environment?

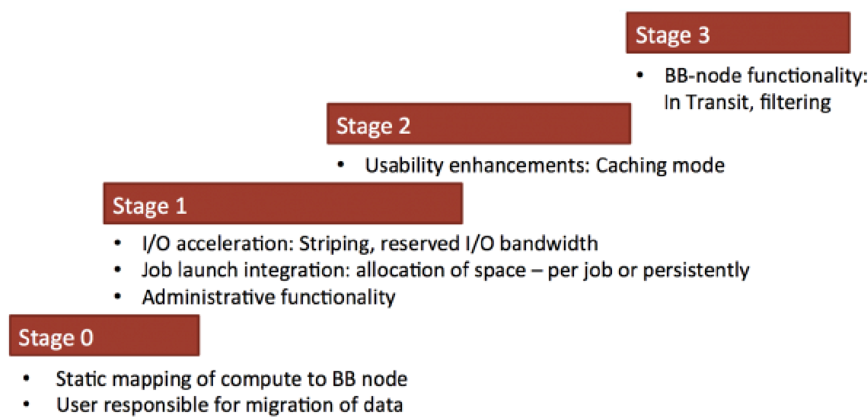


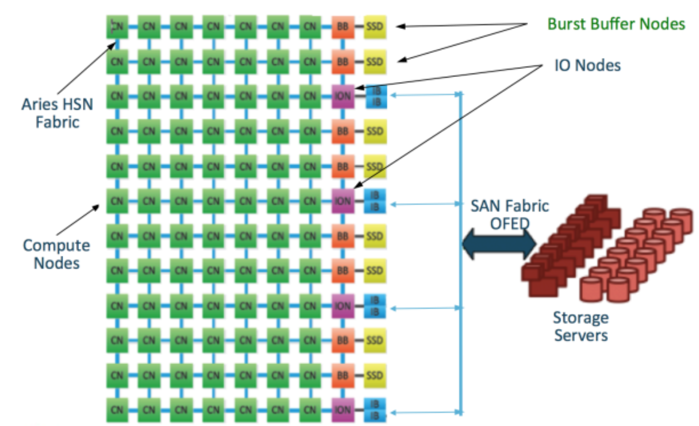
Example visualizations from the Human Connectome Project, CERN/LHC, and the Parallel Ocean Program





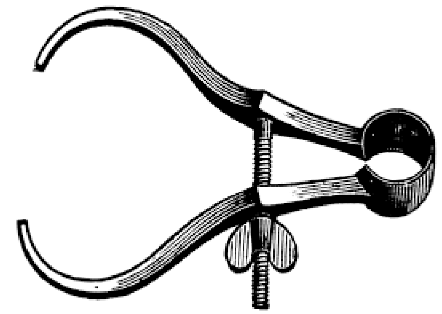
The NERSC burst buffer roadmap and architecture, including solid state burst buffers that can be used in a variety of ways



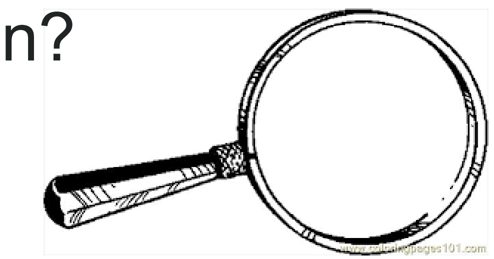


KEY CHALLENGES

- **Instrumentation**:
  - What do we measure?
  - How much overhead is acceptable and when?



- **Analysis**:
  - How do we correlate data and extract actionable information?
  - Can we identify the root cause of performance problems?



- **Impact:**
  - Develop best practices and tune applications
  - Improve system software
  - Design and procure better systems

CHARACTERIZING APPLICATION I/O WITH DARSHAN

WHAT IS DARSHAN?

Darshan is a scalable HPC I/O characterization tool. It captures an accurate but concise picture of ***application*** I/O behavior with minimum overhead.

- No code changes, easy to use
  - Negligible performance impact: just “leave it on”
  - Enabled by default at ALCF, NERSC, and NCSA
  - Used on a case-by-case basis at many other sites
- Produces a ***summary*** of I/O activity for each job, including:
  - Counters for file access operations
  - Time stamps and cumulative timers for key operations
  - Histograms of access, stride, datatype, and extent sizes
- Use cases: observe and tune individual applications, or capture a broad view of the platform workload

DARSHAN DESIGN PRINCIPLES

- The Darshan run time library is inserted at link time (for static executables) or at run time (for dynamic executables)
- Transparent wrappers for I/O functions collect per-file statistics
- Statistics are stored in bounded memory at each rank
- At shutdown time:
  - Collective reduction to merge shared file records
  - Parallel compression
  - Collective write to a single log file
- No communication or storage operations until shutdown
- Command-line tools are used to post-process log files

DARSHAN ANALYSIS EXAMPLE

Example: Darshan-job-summary.pl produces a 3-page PDF file summarizing various aspects of I/O performance

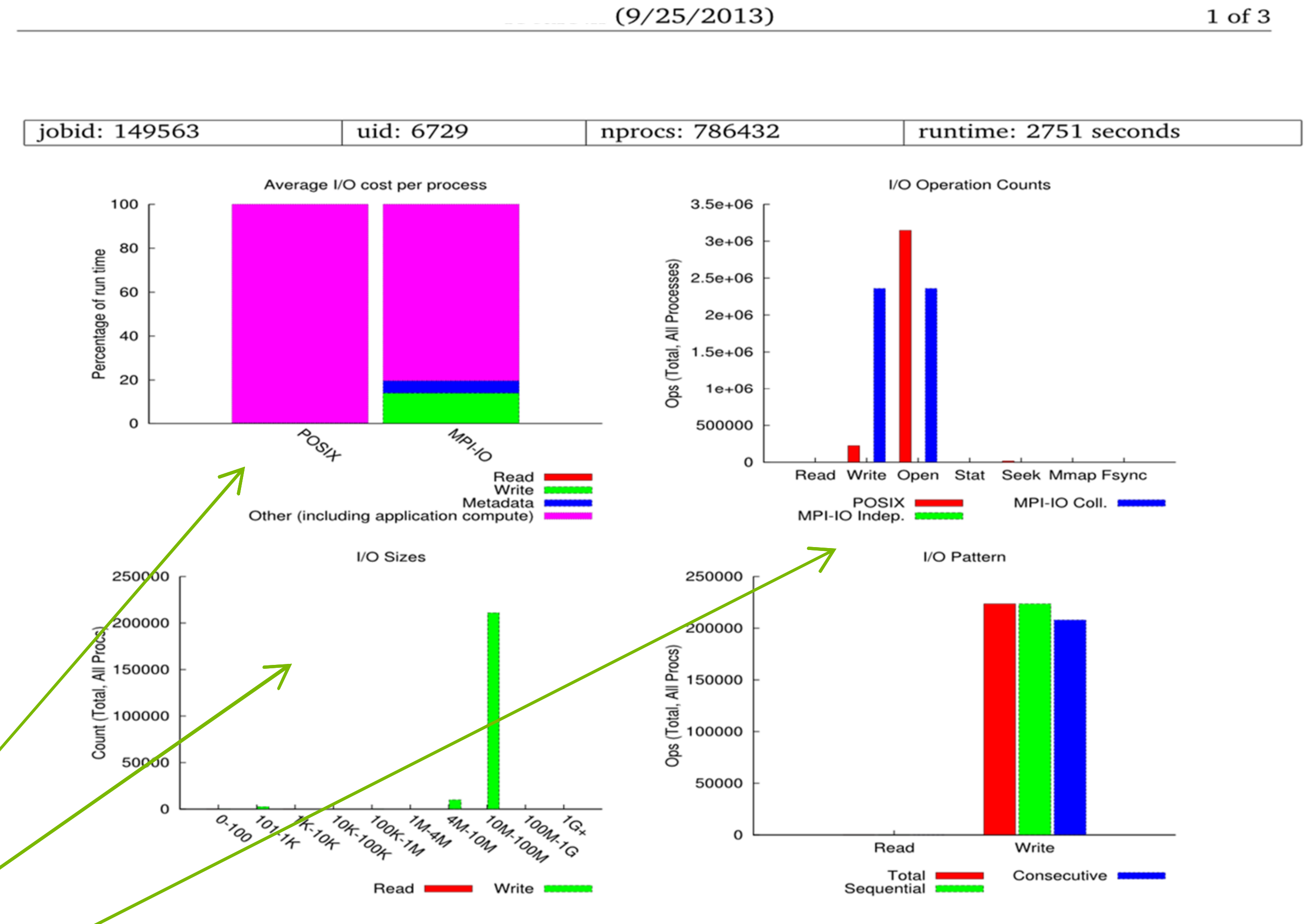
This figure shows the I/O behavior of a 786,432 process turbulence simulation (production run) on the Mira system

Percentage of runtime in I/O

Access size histogram

Access type histograms

File usage



Most Common Access Sizes

access size	count
16777216	210977
8388608	9866
256	2598
68	9

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

DARSHAN DATA MINING EXAMPLE

Use cases for continuous workload characterization

- With a sufficient archive of performance statistics, we can develop heuristics to detect anomalous behavior
- This example highlights large jobs that spent a disproportionate amount of time managing file metadata rather than performing raw data transfer
- Worst offender spent 99% of I/O time in open/close/stat/seek
- This identification process is not yet automated; alerts/triggers are needed for greater impact

Example of heuristics applied to a population of production jobs on the Hopper system in 2013:

JOBS IDENTIFIED USING METADATA RATIO METRIC

Thresholds	meta_time / nprocs $>$ 30 s nprocs $\geq$ 192 metadata_ratio $\geq$ 25%
Total jobs analyzed	261,890
Jobs matching metric	252
Unique users matching metric	45
Largest single-job metadata ratio	$>$ 99%

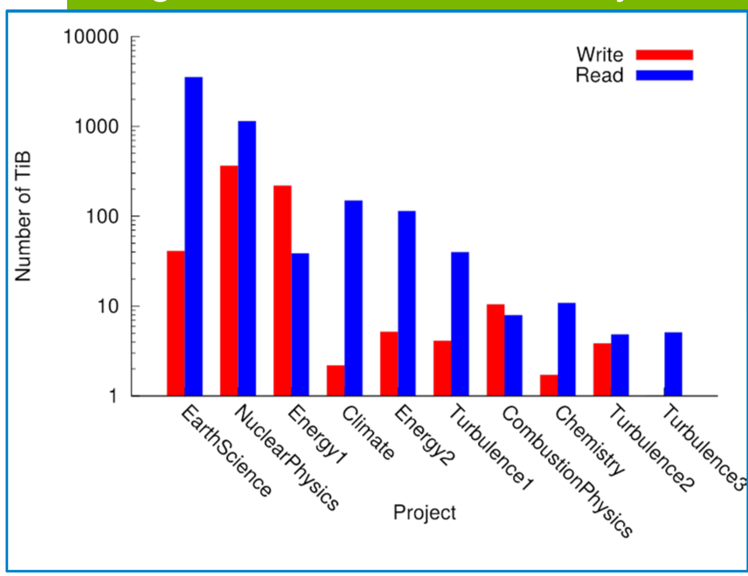
$$\frac{\sum_{n=1}^{nfiles} metadata\_time}{\sum_{n=1}^{nfiles} metadata\_time + IO\_time}$$

Carns et al., "Production I/O Characterization on the Cray XE6," In Proceedings of the Cray User Group meeting 2013 (CUG 2013).

DARSHAN PROJECT TIMELINE

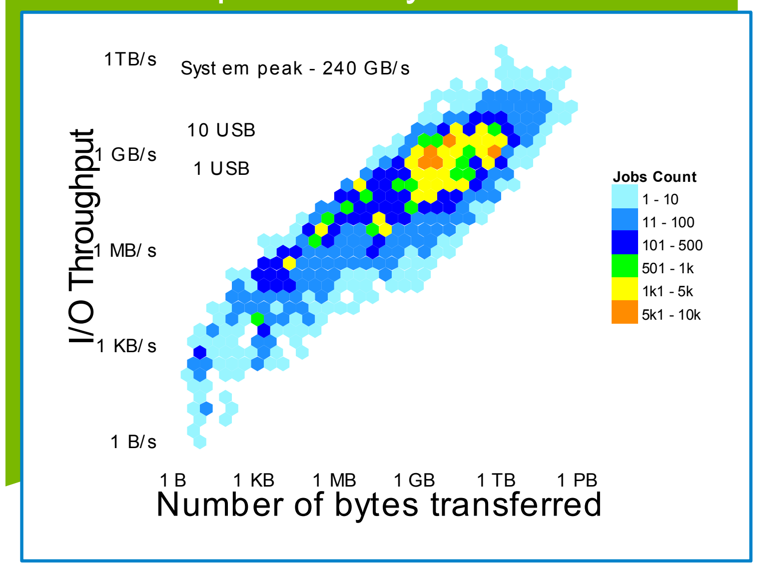
ASCR Base (2008-2011)

- Darshan was conceived to address the need for greater understanding of I/O behavior in diverse scientific applications
- Enabled unprecedented insight into the behavior of the most data-intensive scientific applications at Argonne National Laboratory



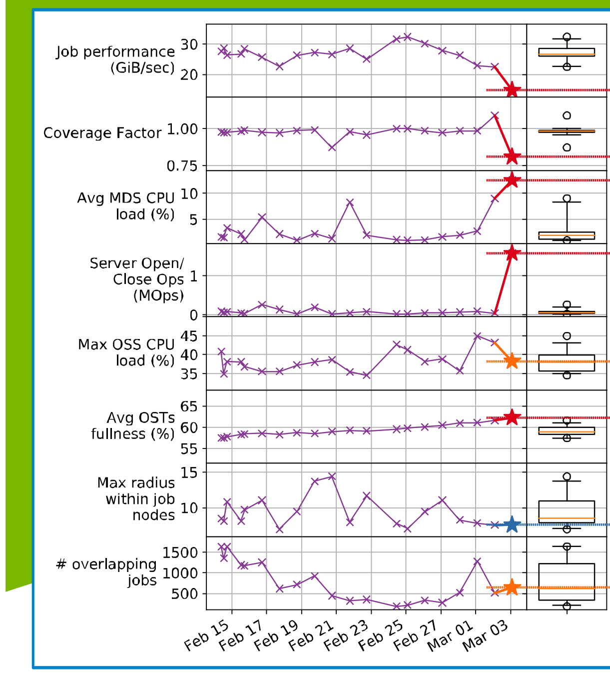
SciDAC (2012-2017)

- Darshan was generalized and ported to multiple computational platforms (IBM BG/Q, Cray XE and XC, Linux clusters) and deployed at every major ASCR facility
- Widespread deployment enabled both cross-platform studies and targeted optimizations to improve scientific productivity



Impact Going Forward

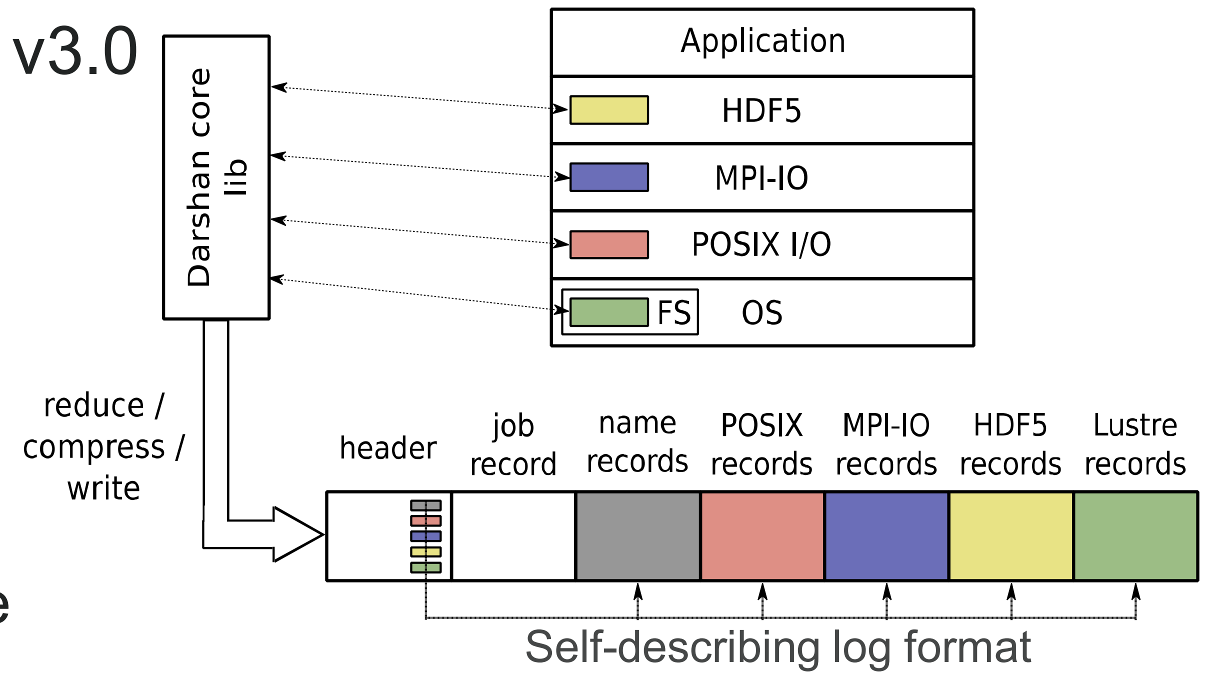
- Darshan is supported by the ALCF, NERSC, OLCF, and NCSA facilities on their largest systems
- Vendors are contributing features
- Darshan is being leveraged in new projects



WHAT'S NEW? MODULARIZED INSTRUMENTATION

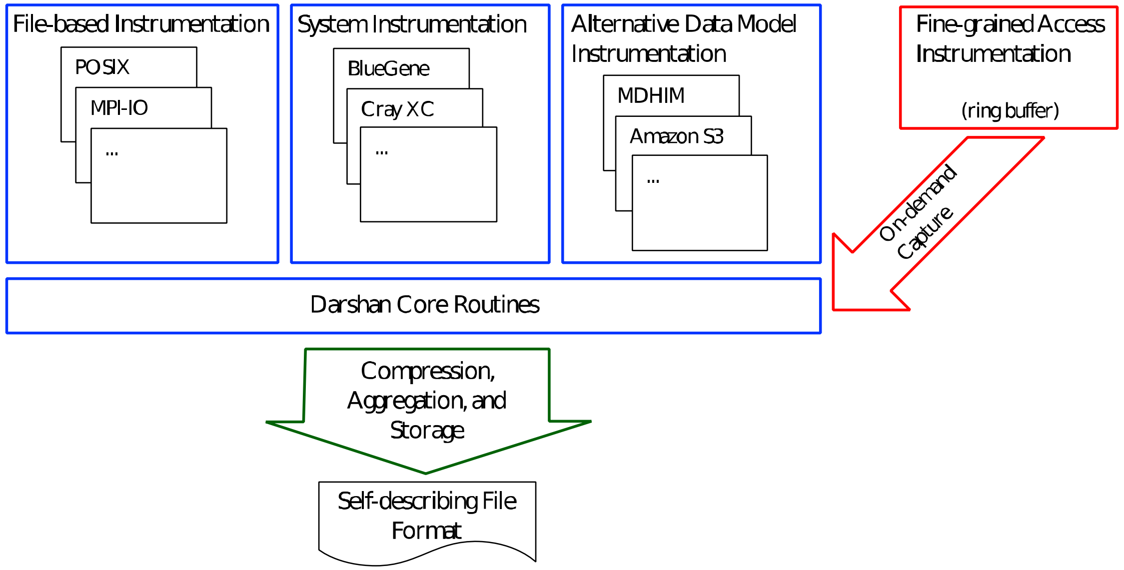
Snyder et al. Modular HPC I/O Characterization with Darshan. In *Proceedings of 5th Workshop on Extreme-scale Programming Tools (ESPT 2016)*, 2016.

- Frequently asked question:
  Can I add instrumentation for X?
- Darshan has been re-architected as a modular framework to help facilitate this, starting in v3.0
- Instrumentation modules:
  - Instruments a source of I/O data (I/O API, file system, etc.)
  - Creates/registers/updates data records characterizing application I/O workload
- Darshan core library:
  - Exposes interface for modules to coordinate
  - Compresses and writes module data to log



DARSHAN MODULE EXAMPLE

- We are using the modular framework to integrate more data sources and simplify the connections between various components in the stack

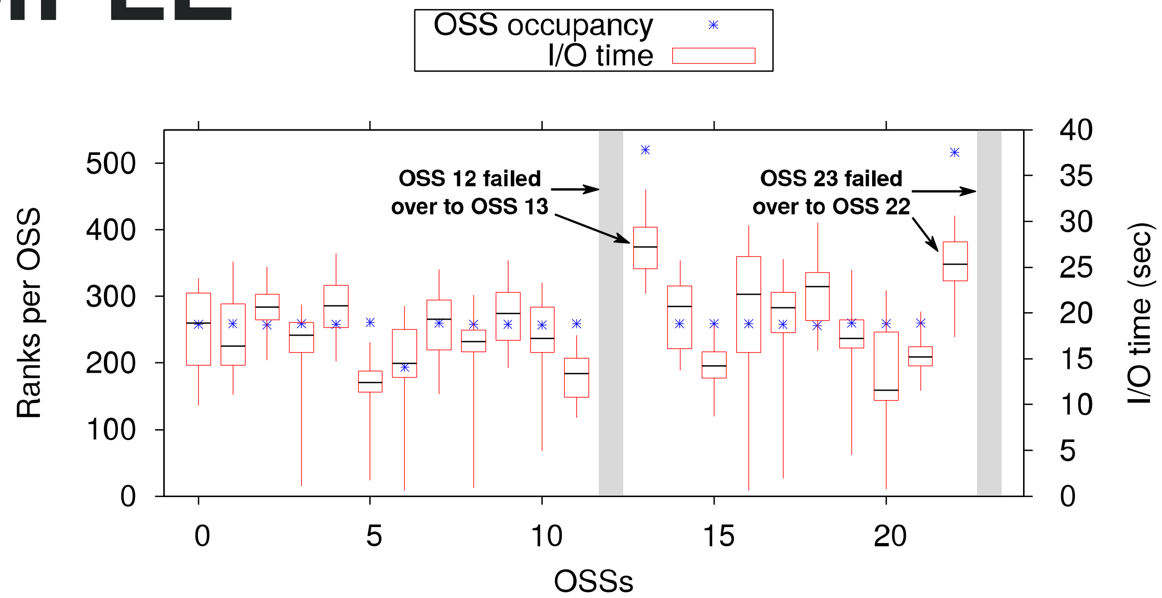


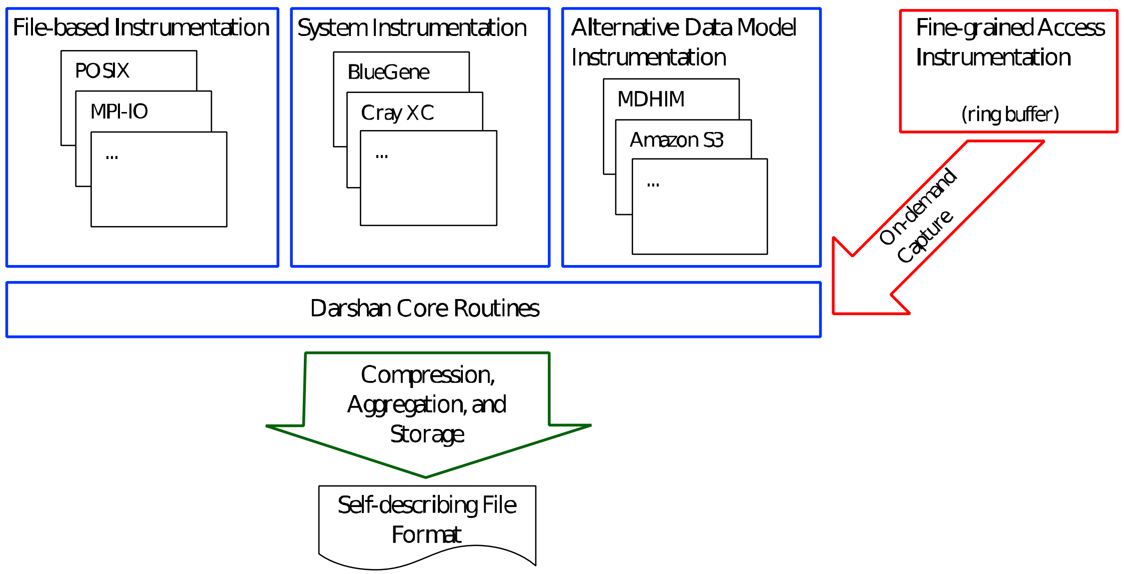
Argonne NATIONAL LABORATORY

DARSHAN MODULE EXAMPLE

- We are using the modular framework to integrate more data sources and simplify the connections between various components in the stack

Per-server I/O load on Edison for a 6,144 process simulation





- Enables new insights into the link between application behavior and system behavior
- In this case we observe how application I/O maps to Lustre file servers
- Combined with system logs, this revealed a bottleneck caused by failover

Argonne NATIONAL LABORATORY

WHAT'S NEW?
DARSHAN EXTENDED TRACING (DXT)

- DXT module provides optional fine-grained instrumentation without recompiling
  - Detailed tracing of POSIX & MPI-IO interfaces
  - Enabled at runtime using environment variable
  - Individual I/O accesses can be mapped to specific Lustre OSTs
- First available in Darshan 3.1.3
- Contributed by Cong Xu and Intel's High Performance Data Division (HPDD)

```
# DXT, file_id: 11616430107429575973, file_name: /home/shane/software/benchmarks/ior/testFile
# DXT, rank: 0, write_count: 4, read_count: 4
# Module    Rank  Wt/Rd  Segment          Offset       Length    Start(s)      End(s)
 X_POSIX       0  write        0               0       262144      0.0067      0.0072
 X_POSIX       0  write        1          262144       262144      0.0072      0.0083
 X_POSIX       0  write        2          524288       262144      0.0083      0.0089
 X_POSIX       0  write        3          786432       262144      0.0089      0.0096
 X_POSIX       0   read        0               0       262144      0.0121      0.0122
 X_POSIX       0   read        1          262144       262144      0.0122      0.0123
 X_POSIX       0   read        2          524288       262144      0.0123      0.0124
 X_POSIX       0   read        3          786432       262144      0.0124      0.0125
```

Xu et al. DXT: Darshan eXtended Tracing. In Proceedings of the Cray User Group meeting 2017 (CUG 2017). TO APPEAR

WHAT'S NEW?
DARSHAN EXTENDED TRACING (DXT)

- Future work in leveraging fine-grain instrumentation capability:
  - When should it be enabled? Dynamic triggers?
  - Can we analyze parts of it at run time?

```
# DXT, file_id: 1161643010742957597З, file_name: /home/shane/software/benchmarks/ior/testFile
# DXT, rank: 0, write_count: 4, read_count: 4
# Module    Rank  Wt/Rd  Segment          Offset       Length    Start(s)      End(s)
 X_POSIX       0  write        0               0       262144      0.0067      0.0072
 X_POSIX       0  write        1          262144       262144      0.0072      0.0083
 X_POSIX       0  write        2          524288       262144      0.0083      0.0089
 X_POSIX       0  write        3          786432       262144      0.0089      0.0096
 X_POSIX       0   read        0               0       262144      0.0121      0.0122
 X_POSIX       0   read        1          262144       262144      0.0122      0.0123
 X_POSIX       0   read        2          524288       262144      0.0123      0.0124
 X_POSIX       0   read        3          786432       262144      0.0124      0.0125
```

THE NEED FOR ONGOING CHARACTERIZATION

- We’ve used Darshan to improving application productivity with case studies, application tuning, and user education
- ... But challenges remain:
  – What other factors influence performance?
  – What if the problem is beyond a user’s control?
  – The user population evolves over time; how do we stay engaged?

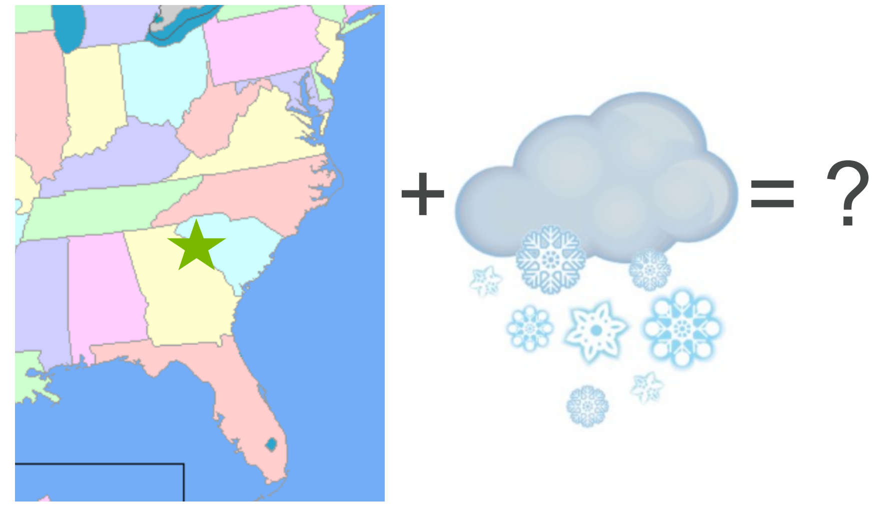
BEYOND APPLICATION CHARACTERIZATION:

HOLISTIC I/O CHARACTERIZATION FOR "TOTAL KNOWLEDGE OF I/O" (TOKIO)

“I OBSERVED PERFORMANCE XYZ.”

Is that normal?

- Consider a climate vs. weather analogy:
  “It is snowing in Atlanta, Georgia.” Is that normal?
- You need context to know:
  - Does it ever snow there?
  - What time of year is it?
  - What was the temperature yesterday?
  - Do your neighbors see snow too?
  - Should you look at it first hand?
- It is similarly difficult to understand a single application performance measurement without broader context
- How do we differentiate typical I/O climate from extreme I/O weather events?



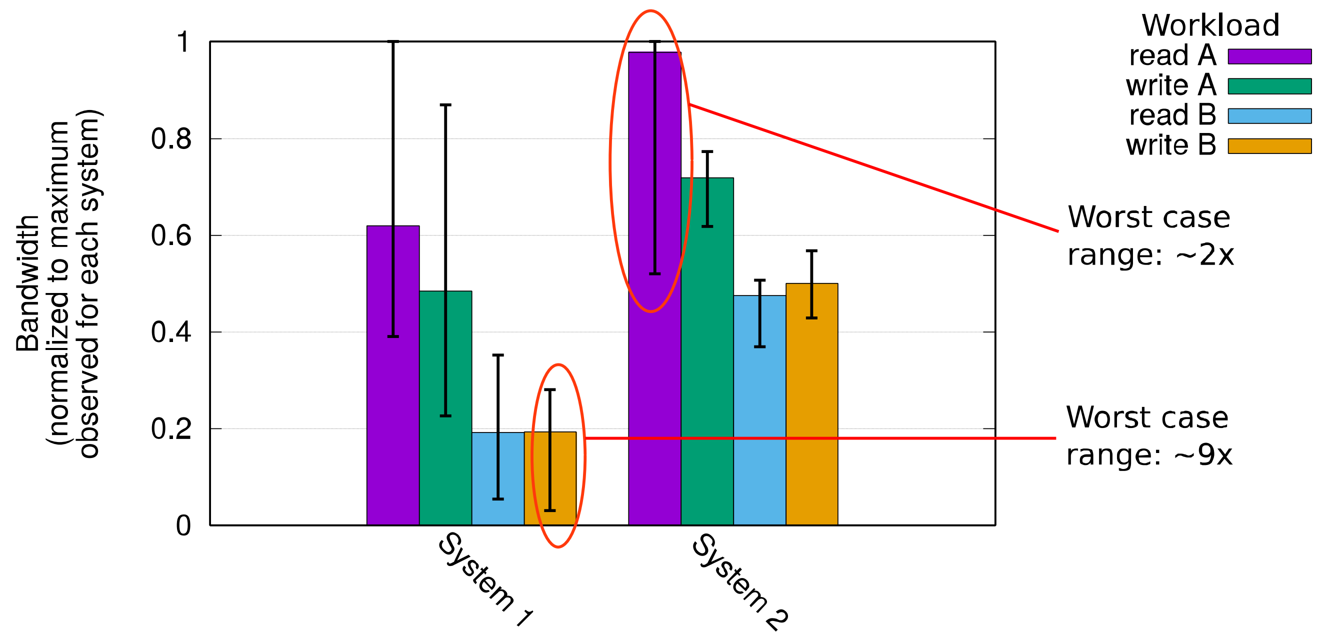
Argonne NATIONAL LABORATORY

HPC I/O VARIABILITY EXAMPLE

Consider the "**I/O fingerprint**" for two example systems on the right

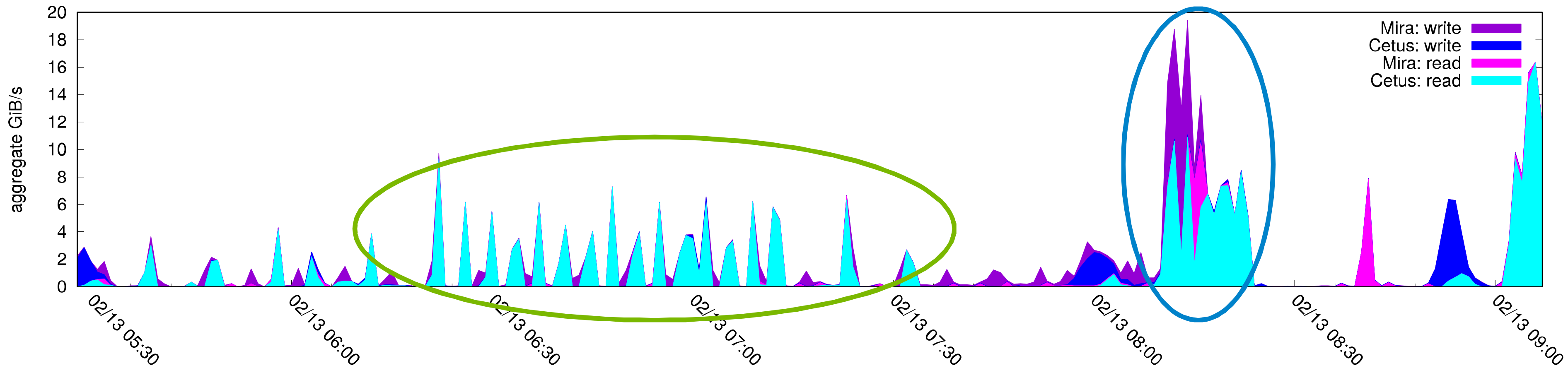
- Data gathered with standardized, periodic, regression sampling of performance for diverse workloads
- More than just a single scalar number for expected I/O performance
- Indicates strengths, weaknesses, and susceptibility to variance

This data also confirms that I/O performance has extraordinary day to day variability.



- This preliminary example show the median performance of several workloads over time on two major computing platforms.
- Performance is normalized to the maximum observed rate on each system, to focus on trends rather than absolute throughput.
- Whiskers indicate minimum and maximum sample values.

WHAT CAUSES VARIABILITY?



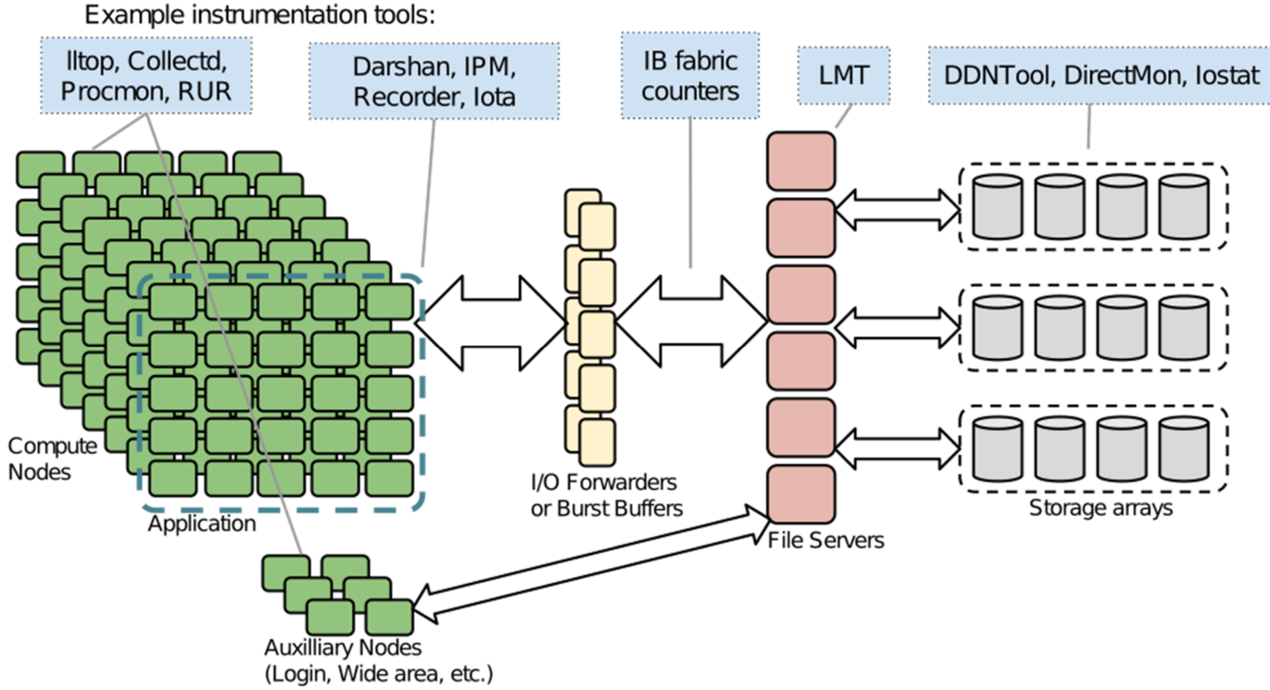
I/O traffic on the ALCF's IBM BG/Q platforms, averaged over one minute intervals. This example window captured individual traffic bursts, regular patterns, and intense read activity.

- The load on the storage system plays is a major factor. HPC I/O workloads are:
  - Inherently bursty
  - With intervals of simulation or analysis punctuated intervals of data movement
- HPC facilities also optimize utilization and reduce cost by sharing major I/O systems across computing systems
- What about network contention, faults, etc.? *There may be many factors!*

CHARACTERIZING THE I/O SYSTEM

What were the contributing factors to a job's performance?

- We need a big picture view
- No lack of instrumentation methods for system components…
  – but with wildly divergent data formats, resolutions, and scope

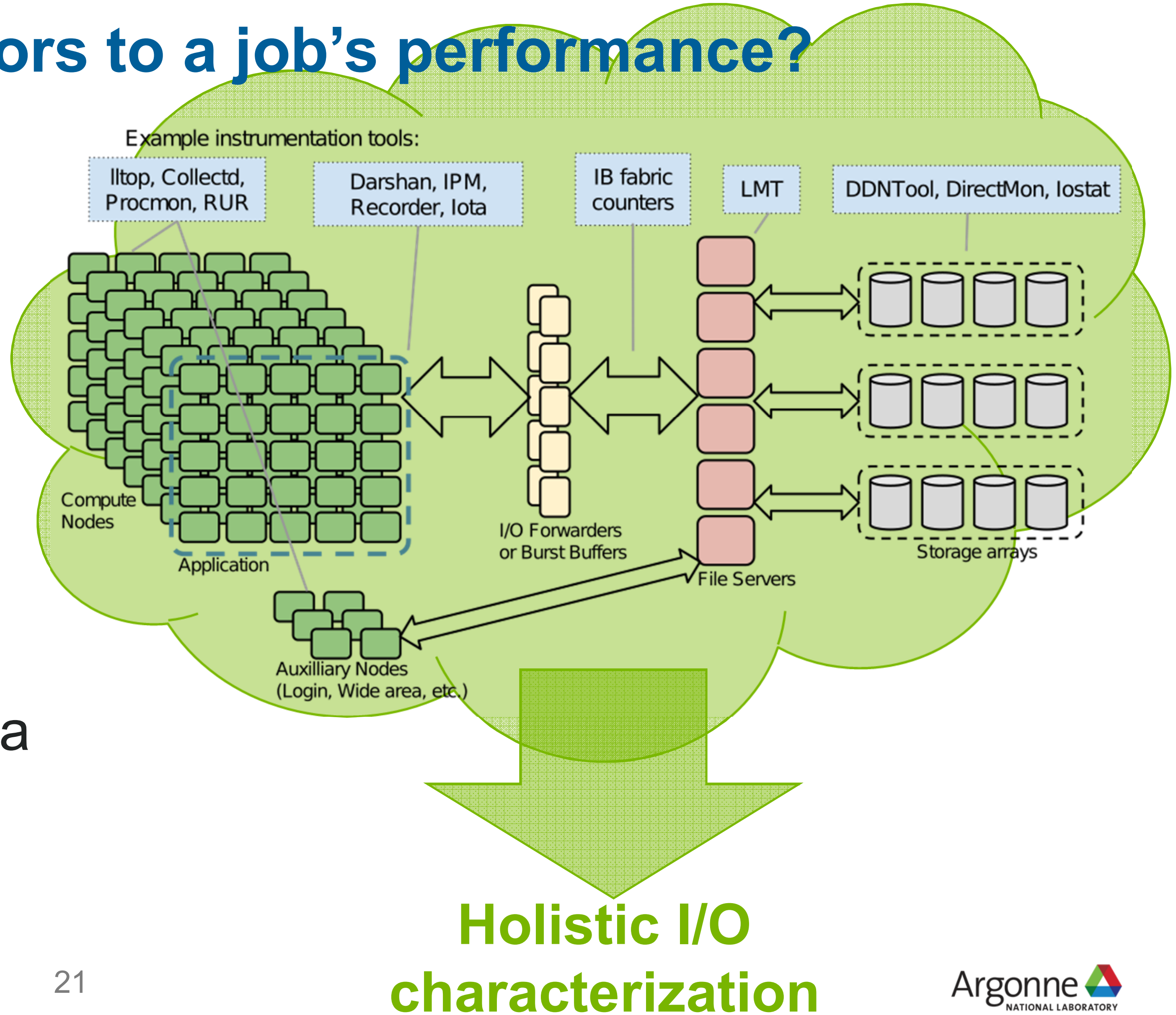


https://www.nersc.gov/research-and-development/tokio/
Collaboration between LBL and ANL
PI: Nick Wright

CHARACTERIZING THE I/O SYSTEM

What were the contributing factors to a job's performance?

- We need a big picture view
- No lack of instrumentation methods for system components…
  - but with wildly divergent data formats, resolutions, and scope
- This is the motivation for the **TOKIO** (TOtal Knowledge of I/O) project:
  - Integrate, correlate, and analyze I/O behavior from the system as a whole for holistic understanding
  - Leverage and extend existing technology where possible



Argonne NATIONAL LABORATORY

TOKIO DEPLOYMENT

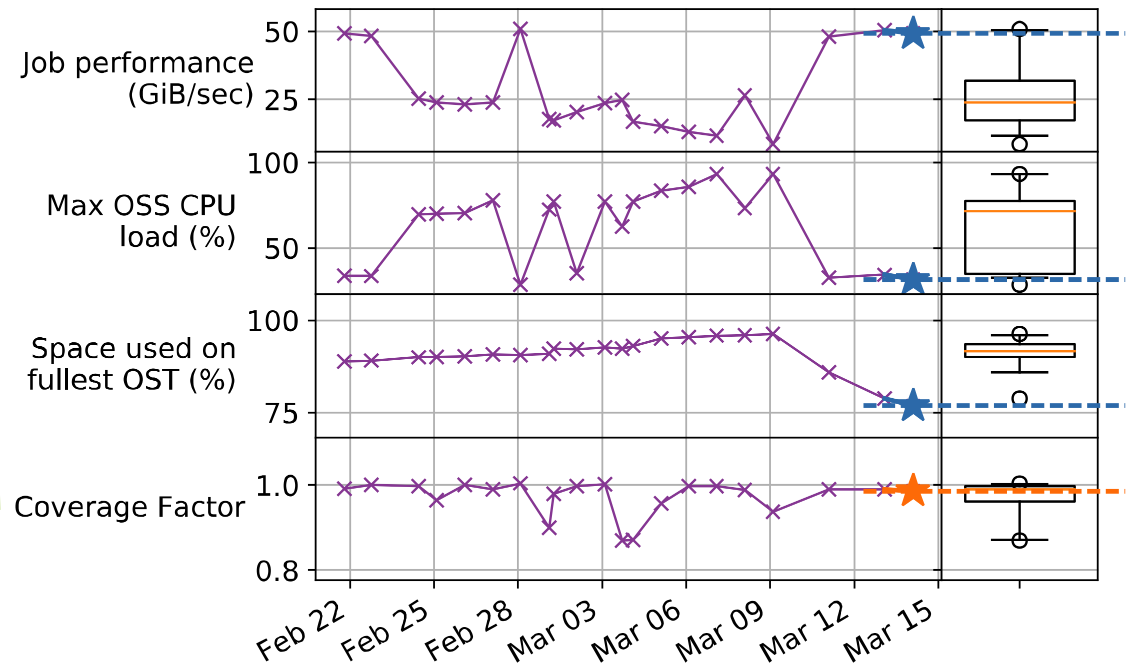
- No new database and (probably) no new instrumentation tool either
- Index and query existing data sources in their native format
  - Infrastructure to align and link data sets
  - Adapters/parsers to produce coherent views on demand
  - Not all systems will have the same data
- Tools sharing a common format for analysis
  - correlation, data mining, dashboards, etc.
- Example of analysis tool: UMAMI

UMAMI

TOKIO Unified Measurements And Metrics Interface

- UMAMI is a pluggable dashboard that displays the I/O performance of an application in context with system telemetry and historical records

Historical samples (for a given application) are plotted over time



Each metric is shown in a separate row

Box plots relate current values to overall variance

(figures courtesy of Glenn Lockwood, NERSC)

Argonne NATIONAL LABORATORY

UMAMI

TOKIO Unified Measurements And Metrics Interface

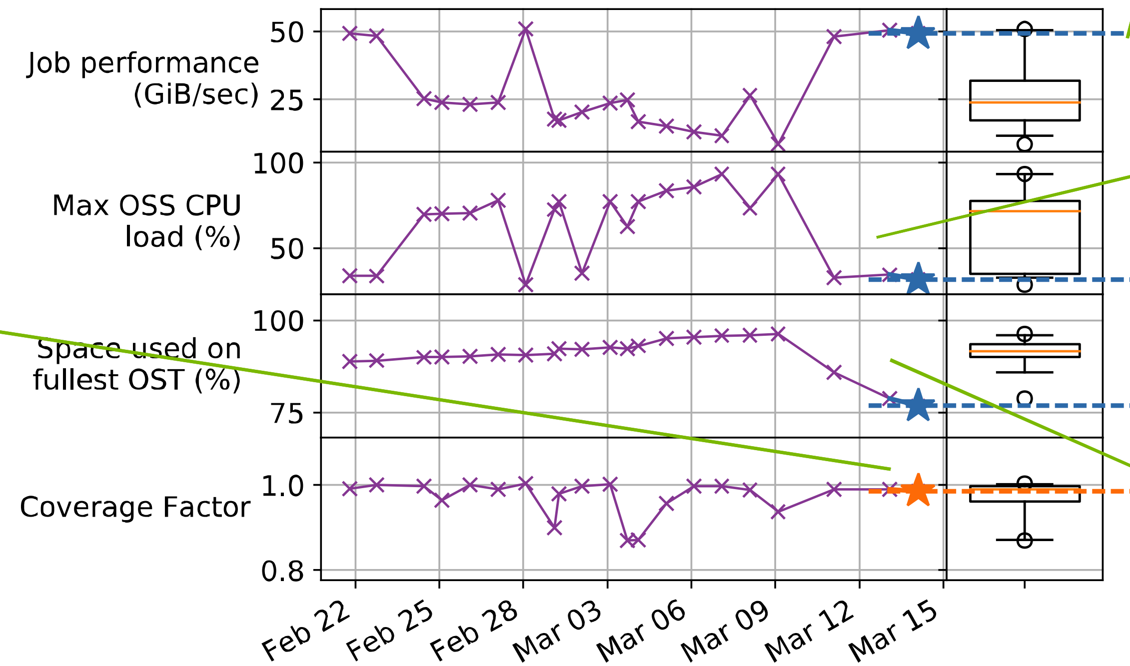
- Broader contextual clues simplify interpretation of unusual performance measurements

Performance for this job is higher than usual

Server CPU load is low after a long-term steady climb

System background load is typical

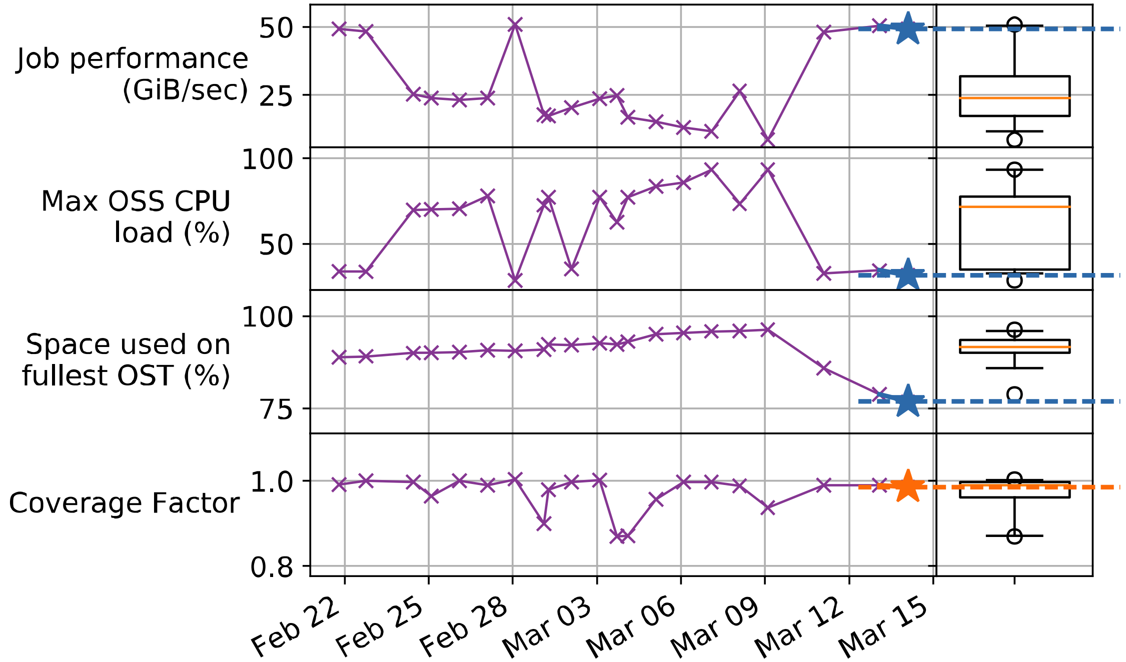
Corresponds to data purge that freed up disk blocks



UMAMI

TOKIO Unified Measurements And Metrics Interface

- UMAMI still requires some level of expert interpretation
- Can we automate the analysis?
  – Automatic root cause determination with Spark
  – Mathematical methods for quantifying correlations and variance

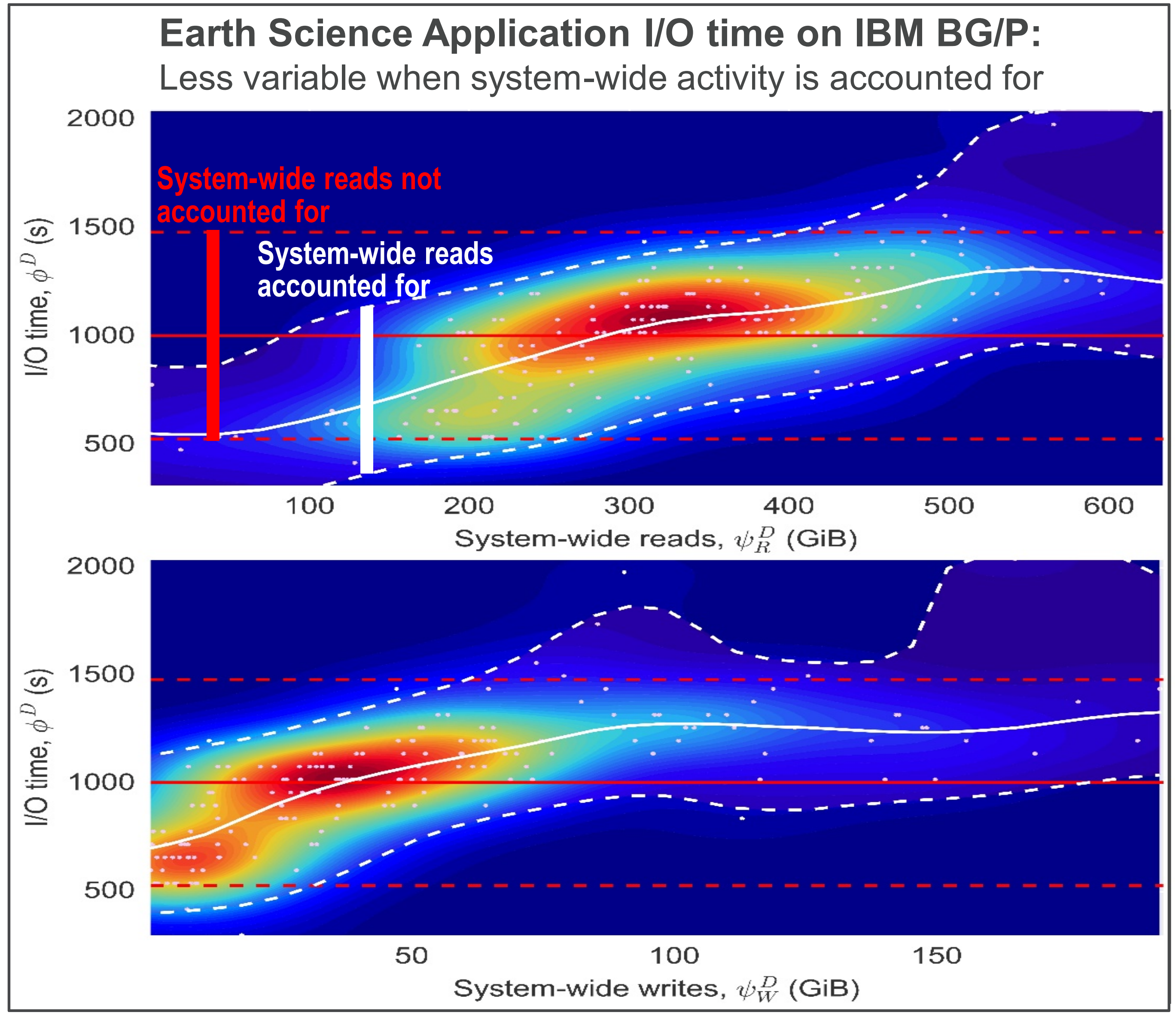


Argonne NATIONAL LABORATORY

MODEL-BASED I/O OPTIMIZATION

We can also leverage holistic I/O instrumentation to develop broader metrics and methodology to understand parallel I/O variability

– Combines SDAV expertise on large-scale I/O with SUPER expertise on performance modeling and multi-metric inference
– Uses server-side and application-level monitoring to analyze job times of prototypical I/O-heavy applications on a BlueGene system at Argonne
– Previous studies ignored effect of parallel system state and workloads



Madireddy et al. Analysis and Correlation of Application I/O Performance and System-Wide I/O Activity. Preprint ANL/MCS-P7036-0417, 2017.

Argonne NATIONAL LABORATORY
SDAV Scalable Data Management, Analysis, and Visualization
SUPER

TOKIO OUTCOME

- Standardized data formats and analysis tools
- Prototypes running at the ALCF (Mira IBM platform) and NERSC (Cori and Edison Cray platforms)
- Software components will be released open source as they are cleaned up from initial prototype
- Data repositories will be published as well
  – This is a secondary but important impact for TOKIO: enabling better data sharing between facilities for comparative studies and best practice development

MORE INFORMATION

- Darshan: http://www.mcs.anl.gov/research/projects/darshan
- TOKIO: http://www.nersc.gov/research-and-development/tokio/

carns@mcs.anl.gov

Argonne NATIONAL LABORATORY

THANK YOU!

THIS WORK WAS SUPPORTED BY THE U.S. DEPARTMENT OF ENERGY, OFFICE OF SCIENCE, ADVANCED SCIENTIFIC COMPUTING RESEARCH, UNDER CONTRACT DE-AC02-06CH11357.

THIS RESEARCH USED RESOURCES OF THE ARGONNE LEADERSHIP COMPUTING FACILITY, WHICH IS A DOE OFFICE OF SCIENCE USER FACILITY SUPPORTED UNDER CONTRACT DE-AC02-06CH11357.

THIS RESEARCH USED RESOURCES OF THE NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER, A DOE OFFICE OF SCIENCE USER FACILITY SUPPORTED BY THE OFFICE OF SCIENCE OF THE U.S. DEPARTMENT OF ENERGY UNDER CONTRACT NO. DE-AC02-05CH11231.