HPC I/O for Computational Scientists: Understanding I/O

Presented to
ATPESC 2017 Participants

Rob Latham and Phil Carns
Mathematics and Computer Science Division
Argonne National Laboratory

Q Center, St. Charles, IL (USA)
8/4/2017

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

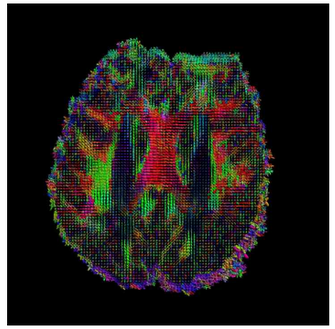
Argonne NATIONAL LABORATORY

Motivation for Characterizing parallel I/O

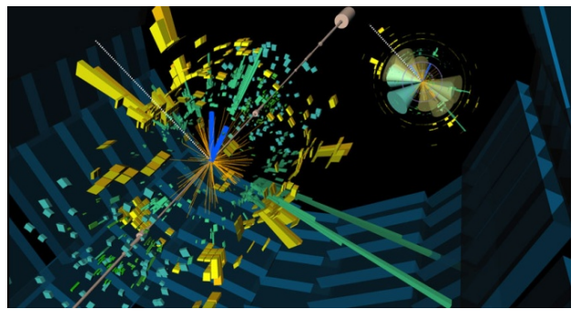
Times are changing in HPC storage!

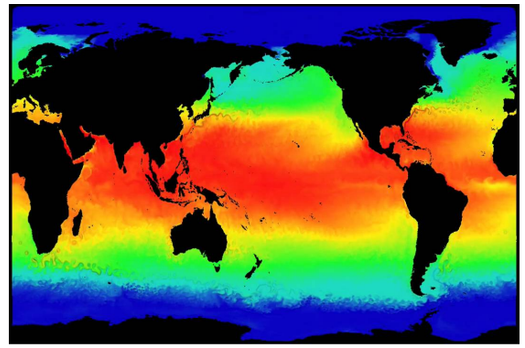
- Most scientific domains are increasingly data intensive: climate, physics, biology and much more
- Upcoming platforms include complex hierarchical storage systems

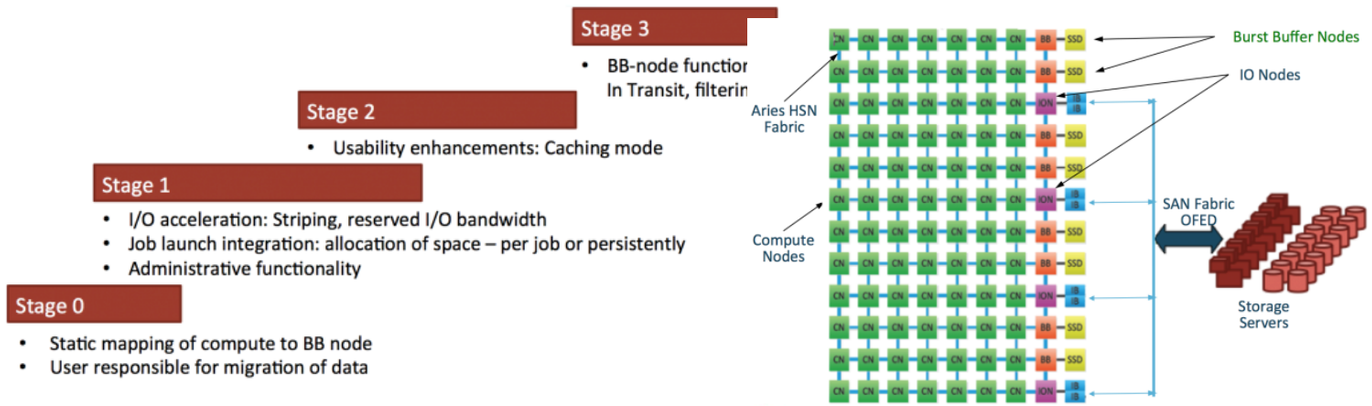
How can we maximize productivity in this environment?



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







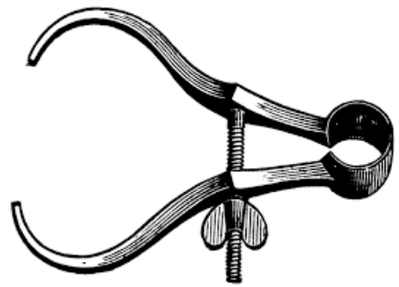
The NERSC burst buffer roadmap and architecture, including solid state burst buffers that can be used in a variety of ways

Argonne NATIONAL LABORATORY

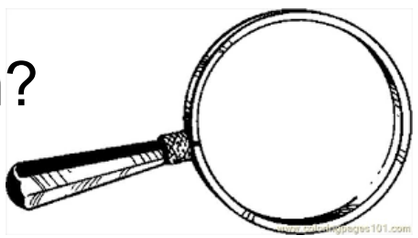
ECP EXASCALE COMPUTING PROJECT

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

Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

CHARACTERIZING APPLICATION I/O WITH DARSHAN

Argonne NATIONAL LABORATORY

What is Darshan?

Project began in 2008, first public software release and deployment in 2009

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, NCSA, and KAUST
  - Installed and available for case by case use 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

Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

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

JOB analysis example

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

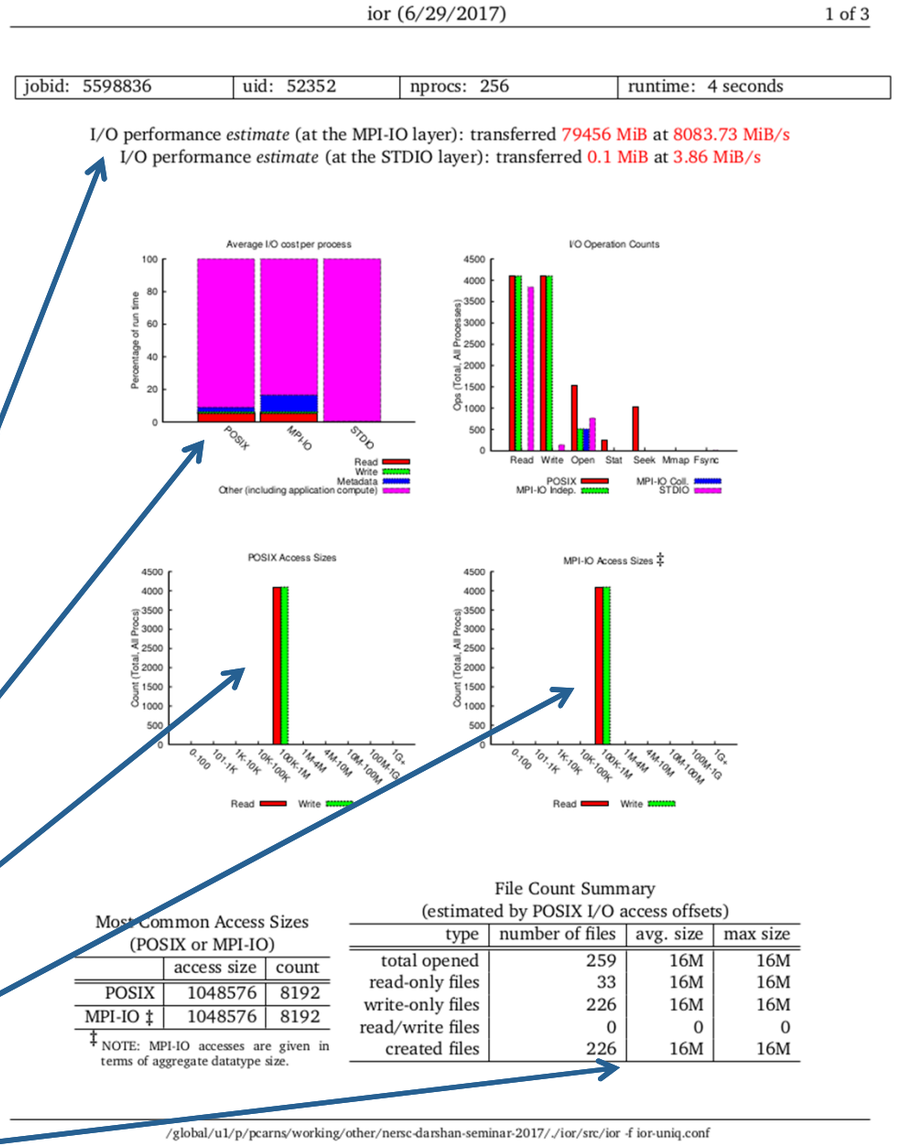
Estimated performance

Percentage of runtime in I/O

Access size histogram

Access type histograms

File usage



ior (6/29/2017) 1 of 3

jobid: 5598836	uid: 52352	nprocs: 256	runtime: 4 seconds

I/O performance *estimate* (at the MPI-IO layer): transferred 79456 MiB at 8083.73 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.1 MiB at 3.86 MiB/s

Most Common Access Sizes (POSIX or MPI-IO)

	access size	count
POSIX	1048576	8192
MPI-IO ‡	1048576	8192

‡ NOTE: MPI-IO accesses are given in terms of aggregate datatype size.

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

type	number of files	avg. size	max size
total opened	259	16M	16M
read-only files	33	16M	16M
write-only files	226	16M	16M
read/write files	0	0	0
created files	226	16M	16M

/global/u1/p/pcarns/working/other/nersc-darshan-seminar-2017/./ior/src/ior -f ior-uniq.conf

SYSTEM analysis example

- 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 in future work 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%

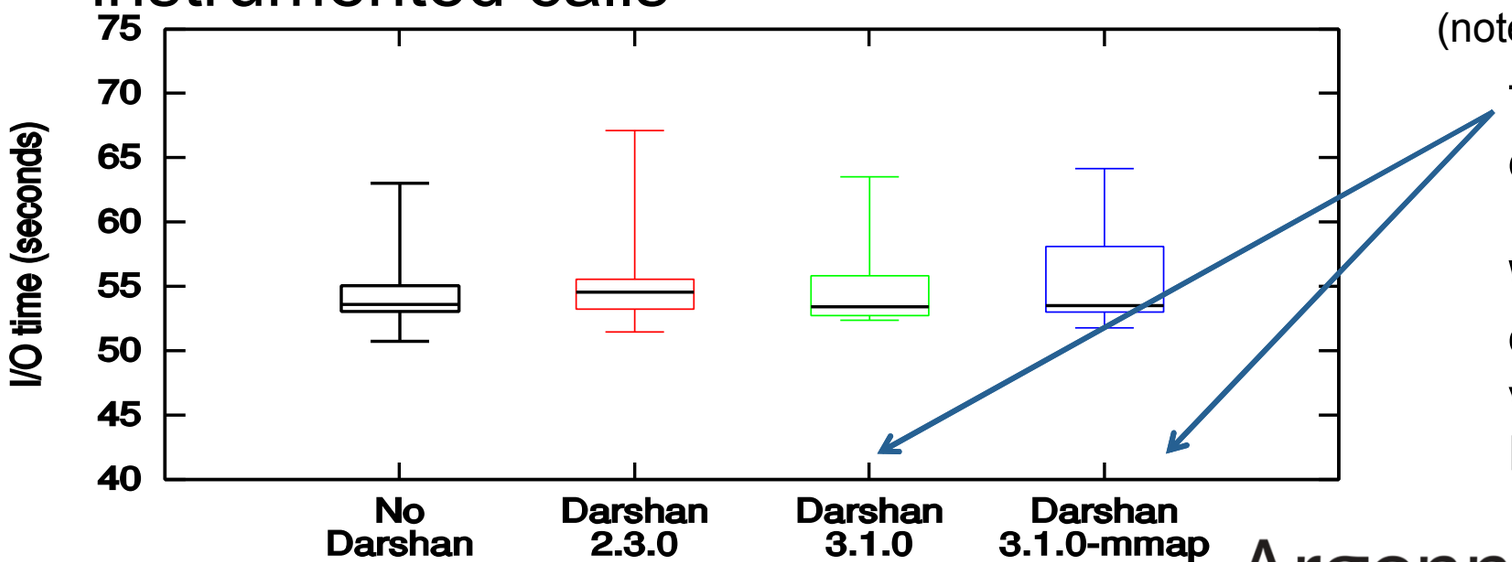
$$\frac{\sum_{n=1}^{nfiles} metadata\_time}{\sum_{n=1}^{nfiles} metadata\_time + IO\_time}$$

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

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

Performance: function wrapping overhead

- What is the cost of interposing Darshan I/O instrumentation wrappers?
- To test, we compare observed I/O time of an IOR configuration linked against different Darshan versions on *Edison*
- File-per-process workload, 6,000 processes, over 12 million instrumented calls



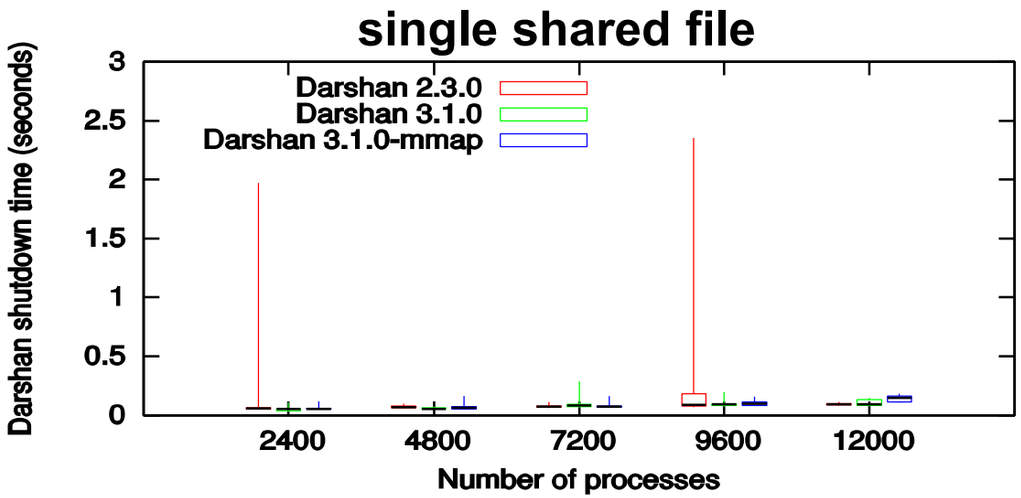
(note that the Y axis labels start at 40)

Type of Darshan builds now deployed on Theta and Cori

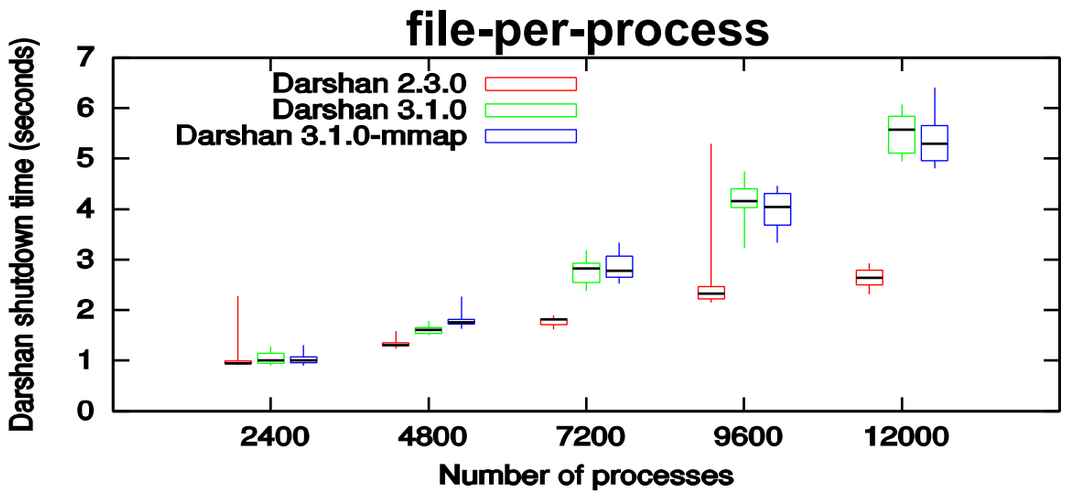
Why the box plots? Recall observation from this morning that variability is a constant theme in HPC I/O today.

Performance: shutdown overhead

- Involves aggregating, compressing, and collectively writing I/O data records
- To test, synthetic workloads are injected into Darshan and resulting shutdown time is measured on *Edison*



Near constant shutdown time of ~100 ms in all cases



Shutdown time scales linearly with job size:

5-6s extra shutdown time with 12,000 files

Argonne NATIONAL LABORATORY

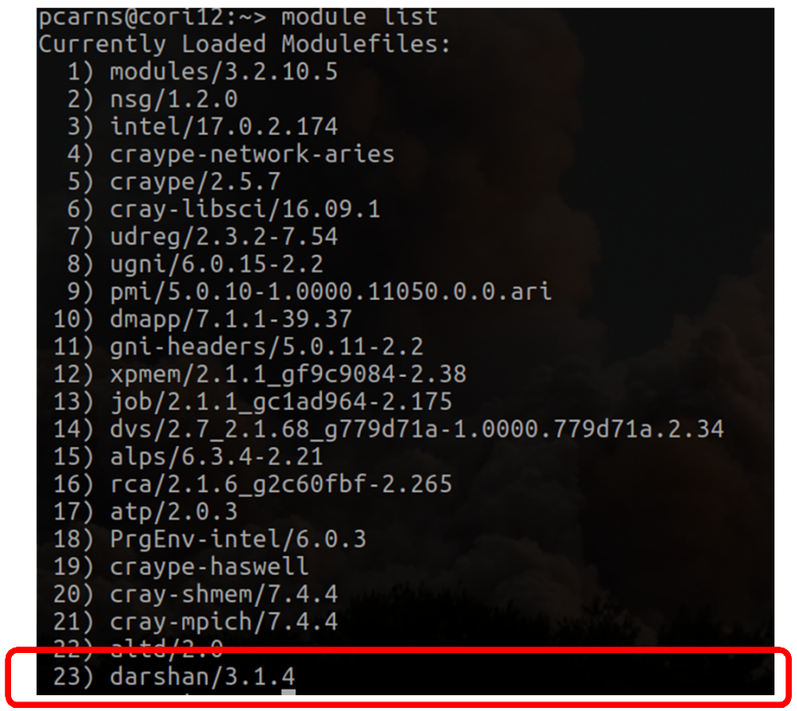
ECP EXASCALE COMPUTING PROJECT

USING DARSHAN IN PRACTICE

Argonne NATIONAL LABORATORY

Typical deployment and usage

- Darshan usage on Mira, Cetus, Vesta, Theta, Cori, or Edison, abridged:
  - Run your job
  - If the job calls MPI_Finalize(), log will be stored in DARSHAN_LOG_DIR/**month**/**day**/
  - Theta: /lus/theta-fs0/logs/darshan/theta
  - Use tools (next slides) to interpret log
- On Titan: "module load darshan" first
- Links to documentation with details will be given at the end of this presentation



Generating job summaries

- Run job and find its log file:

```
pcarns@cori07:~/working/other/nersc-darshan-seminar-2017> sbatch ior-shared.sh
Submitted batch job 5598961
pcarns@cori07:~/working/other/nersc-darshan-seminar-2017> ls /global/cscratch1/s
d/darshanlogs/2017/6/29 |grep 5598961
pcarns_ior_id5598961_6-29-62551-10312342380485257913_1.darshan
```

Job id

Corresponding log file in today's directory

- Copy log file to current directory, generate PDF summary:

```
pcarns@cori12:~/working/other/nersc-darshan-seminar-2017/logs> cp /global/cscrat
ch1/sd/darshanlogs/2017/6/29/*carns* .
pcarns@cori12:~/working/other/nersc-darshan-seminar-2017/logs> ls
pcarns_ior_id5598836_6-29-61782-18110051766566701976_1.darshan
pcarns_ior_id5598961_6-29-62551-10312342380485257913_1.darshan
pcarns_ior_id5599243_6-29-62856-98294316949384266666_1.darshan
pcarns_ior_id5599615_6-29-63159-27127056540901360 29_1.darshan
pcarns@cori12:~/working/other/nersc-darshan-seminar-2017/logs> module load latex
pcarns@cori12:~/working/other/nersc-darshan-seminar-2017/logs> darshan-job-summa
ry.pl pcarns_ior_id5598836_6-29-61782-18110051766566701976_1.darshan
```

Copy out logs

List logs

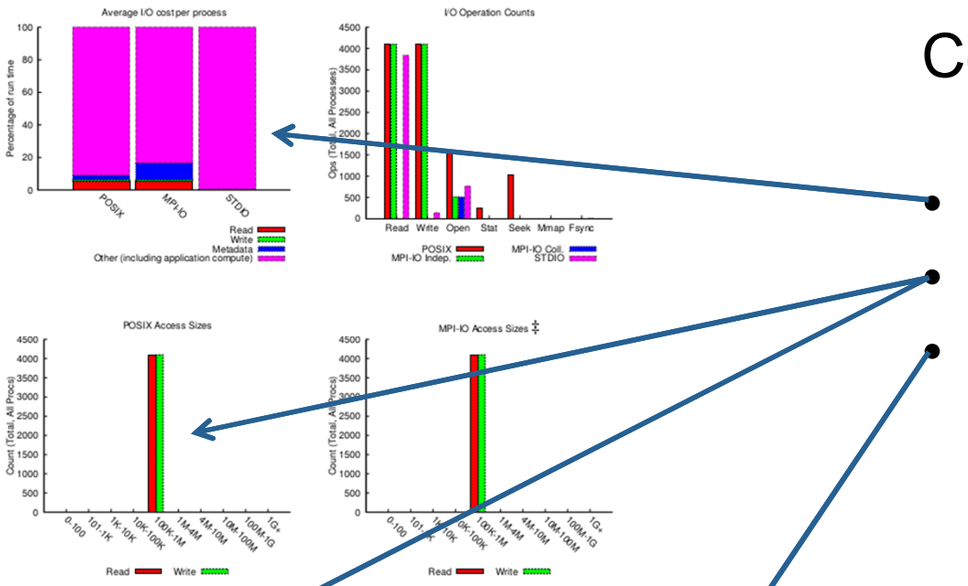
Load "latex" module, (if needed)

Generate PDF

First page of summary

jobid: 5598836	uid: 52352	nprocs: 256	runtime: 4 seconds

I/O performance *estimate* (at the MPI-IO layer): transferred 79456 MiB at 8083.73 MiB/s
I/O performance *estimate* (at the STDIO layer): transferred 0.1 MiB at 3.86 MiB/s



Most Common Access Sizes
(POSIX or MPI-IO)

	access size	count
POSIX	1048576	8192
MPI-IO ‡	1048576	8192

‡ NOTE: MPI-IO accesses are given in terms of aggregate datatype size.

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

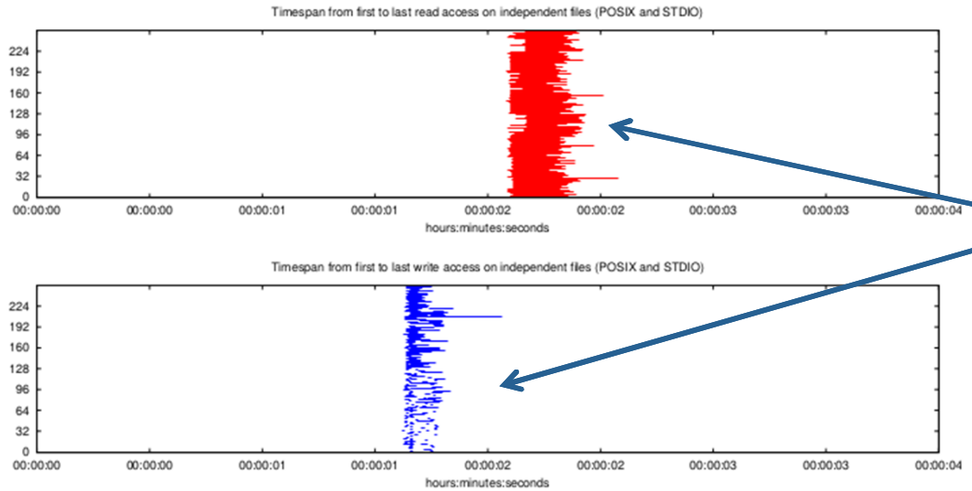
type	number of files	avg. size	max size
total opened	259	16M	16M
read-only files	33	16M	16M
write-only files	226	16M	16M
read/write files	0	0	0
created files	226	16M	16M

/global/u1/p/pcarns/working/other/nersc-darshan-seminar-2017/./ior/src/ior -f ior-uniq.conf

Common questions:

- Did I spend much time performing IO?
- What were the access sizes?
- How many files where opened, and how big were they?

Second page of summary (excerpt)



Common questions:

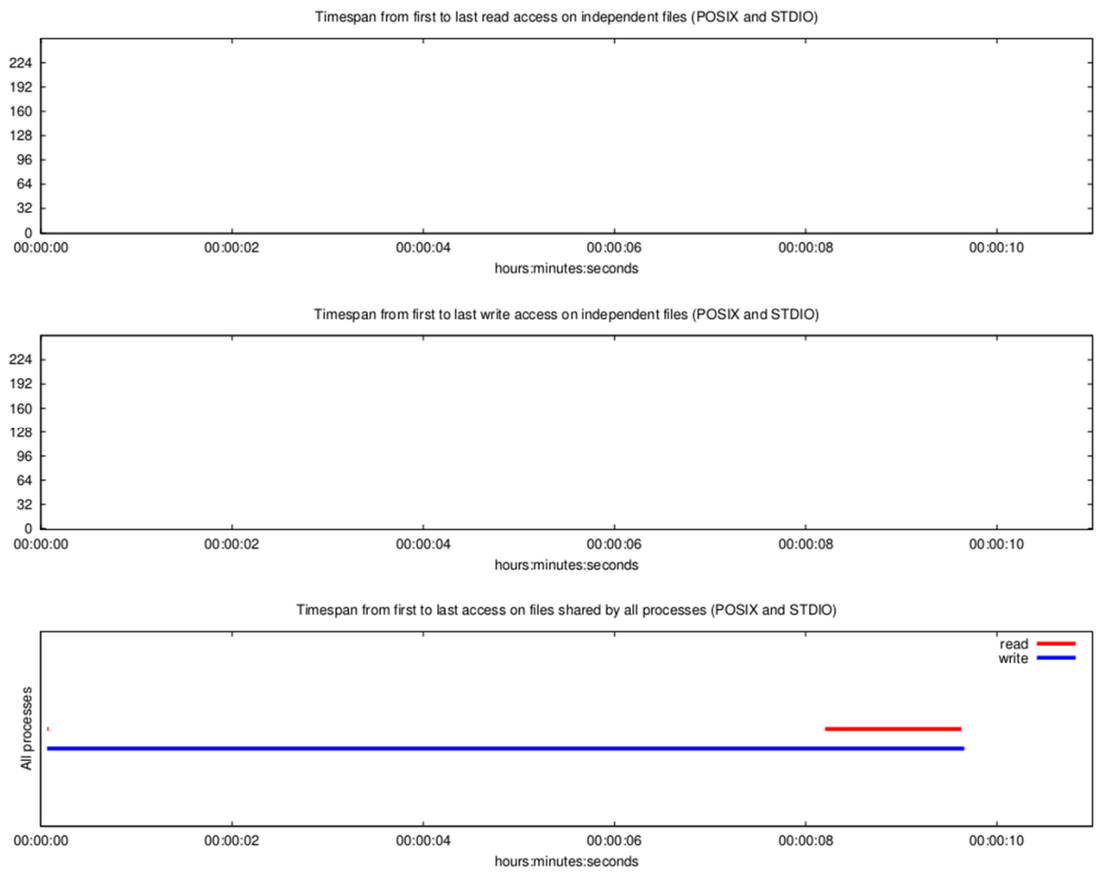
- Where in the timeline of the execution did ea
  rank do I/O?

There are additional graphs in the PDF file with increasingly detailed information.
You can also dump all data from the log in text format using “darshan-parser”.

TIPS AND TRICKS: ENABLING ADDITIONAL DATA CAPTURE

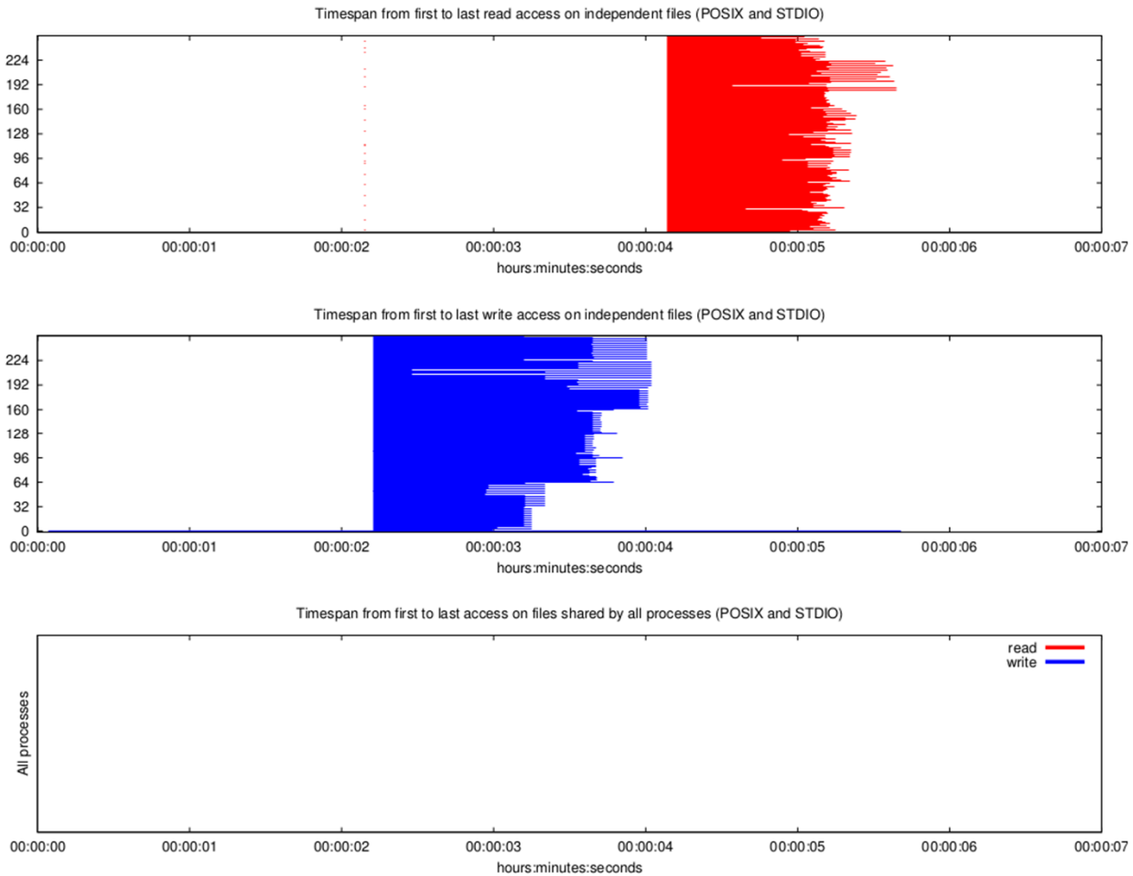
Argonne NATIONAL LABORATORY

What if you are doing shared-file IO?



- Your timeline might look like this
- No per-process information available because the data was aggregated by Darshan to save space/overhead
- Is that important? It depends on what you need to learn about your application.
  - It may be interesting for applications that access the same file in distinct phases over time

What if you are doing shared-file IO?




```
#!/bin/bash -l
#SBATCH -p debug
#SBATCH -A m888
#SBATCH -N 8
#SBATCH -t 0:15:00
#SBATCH -J ior-example
#SBATCH -o ior-example.o%j
#SBATCH --mail-type=BEGIN,END
#SBATCH --mail-user=carns@mcs.anl.gov
#SBATCH -C haswell
#SBATCH -L SCRATCH

export DARSHAN_DISABLE_SHARED_REDUCTION=1

stripe_medium $SCRATCH/ior
srun -n 256 ./ior/src/ior -f ior-shared.conf
```

- Set environment variable to disable shared file reductions
- Increases overhead and log file size, but provides per-rank info even on shared files

Detailed trace data

```
#!/bin/bash -l
#SBATCH -p debug
#SBATCH -A m888
#SBATCH -N 8
#SBATCH -t 0:15:00
#SBATCH -J ior-example
#SBATCH -o ior-example.o%j
#SBATCH --mail-type=BEGIN,END
#SBATCH --mail-user=carns@mcs.anl.gov
#SBATCH -C haswell
#SBATCH -L SCRATCH

export DXT_ENABLE_IO_TRACE=4

stripe_medium $SCRATCH/ior
srun -n 256 ./ior/src/ior -f ior-shared.conf
```

- Set environment variable to enable “DXT” tracing
- This causes additional overhead and larger files, but captures precise access data
- Parse trace with “darshan-dxt-parser”

```
pcarns@cori12:~/working/other/nersc-darshan-seminar-2017/logs> darshan-dxt-parser pcarns_ior_id
5599892_6-29-65443-4368632872761953932_1.darshan
# ************************************************
# DXT_POSIX module data
# ************************************************

# DXT, file_id: 11542722479531699073, file_name: /global/cscratch1/sd/pcarns/ior/ior.dat
# DXT, rank: 0, hostname: nid00511
# DXT, write_count: 16, read_count: 16
# DXT, mnt_pt: /global/cscratch1, fs_type: lustre
# DXT, Lustre stripe_size: 1048576, Lustre stripe_count: 24
# DXT, Lustre OST obdidx: 49 185 115 7 135 3 57 95 43 27 191 1 163 51 15 153 187 55 151 239 79
25 137 47
# Module    Rank  Wt/Rd  Segment          Offset       Length    Start(s)      End(s)  [OST]
 X_POSIX       0  write        0               0      1048576      0.7895      0.8267  [ 49]
 X_POSIX       0  write        1         1048576      1048576      0.8267      0.9843  [185]
 X_POSIX       0  write        2         2097152      1048576      0.9843      1.0189  [115]
 X_POSIX       0  write        3         3145728      1048576      1.0189      1.0250  [   7]
 X_POSIX       0  write        4         4194304      1048576      1.0250      1.0319  [135]
 X_POSIX       0  write        5         5242880      1048576      1.0319      1.0459  [   3]
```

Feature contributed by Cong Xu and Intel’s High Performance Data Division

Cong Xu et. al, "DXT: Darshan eXtended Tracing", Cray User Group Conference 2017

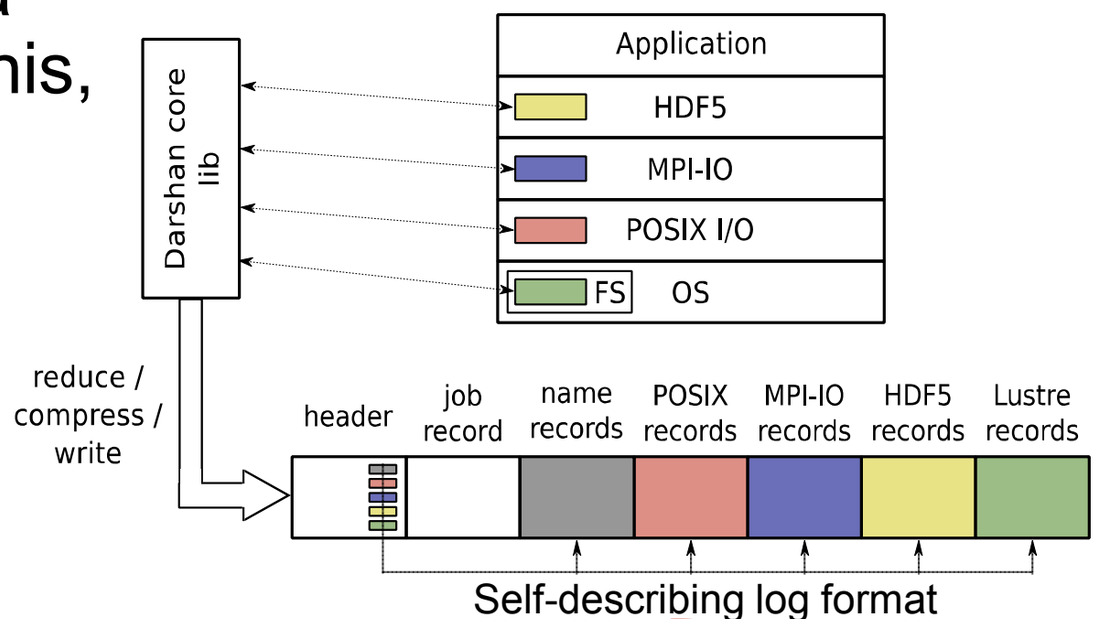
DARSHAN FUTURE WORK

Argonne NATIONAL LABORATORY

What's new?
Modularized instrumentation

- 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

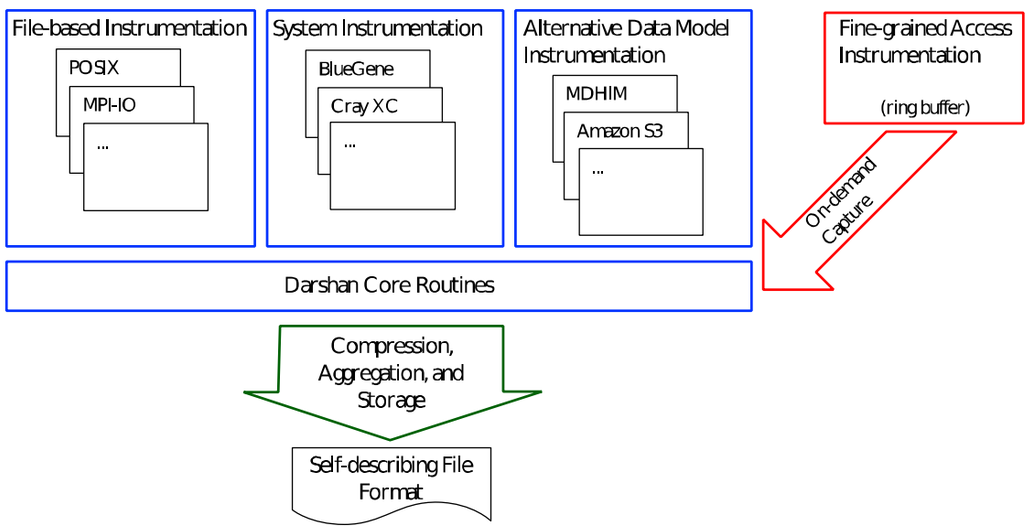
Snyder et al. Modular HPC I/O Characterization with Darshan. In *Proceedings of 5th Workshop on Extreme-scale Programming Tools (ESPT 2016),* 2016.



Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

Darshan Module example



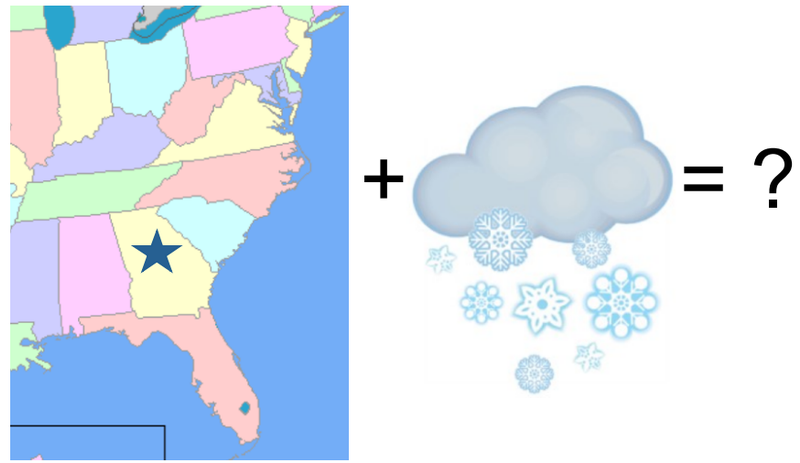
- We are using the modular framework to integrate more data sources and simplify the connections between various components in the stack
- This is a good way for collaborators to get involved in Darshan development

The need for HOLISTIC 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?

“I observed performance XYZ. Now what?”

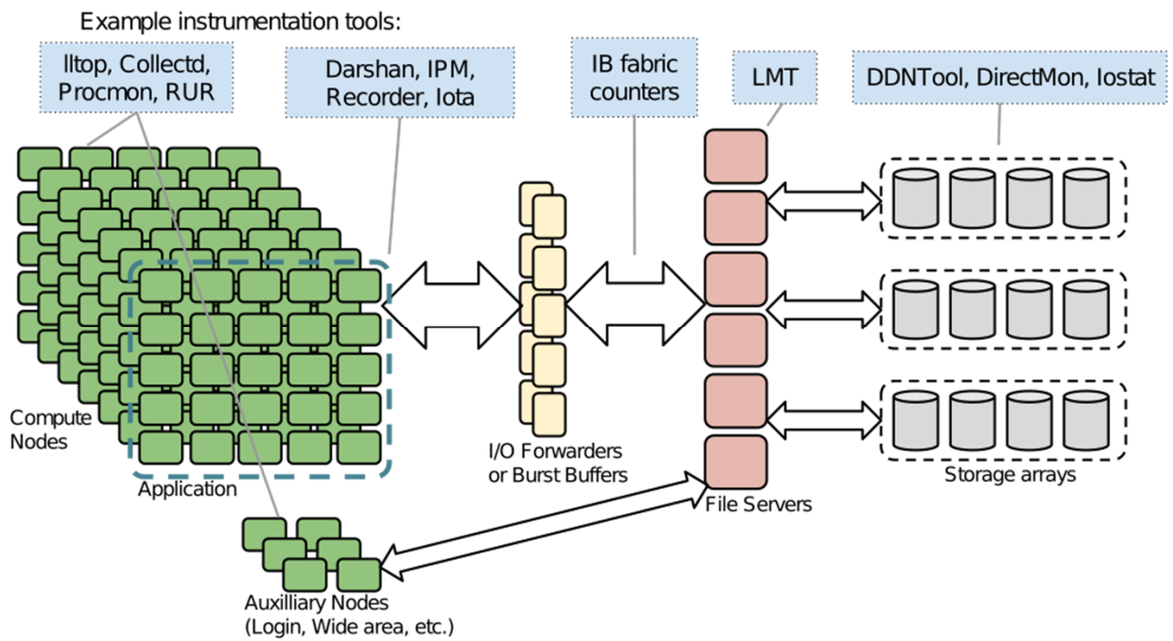
- A climate vs. weather analogy: It is snowing in Atlanta, Georgia. Is that normal?
- You need *context* to know:
  - Does it ever snow there?
  - What time of year is it?
  - What was the temperature yesterday?
  - Do your neighbors see snow too?
  - Should you look at it first hand?



- 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?**

Characterizing the I/O system

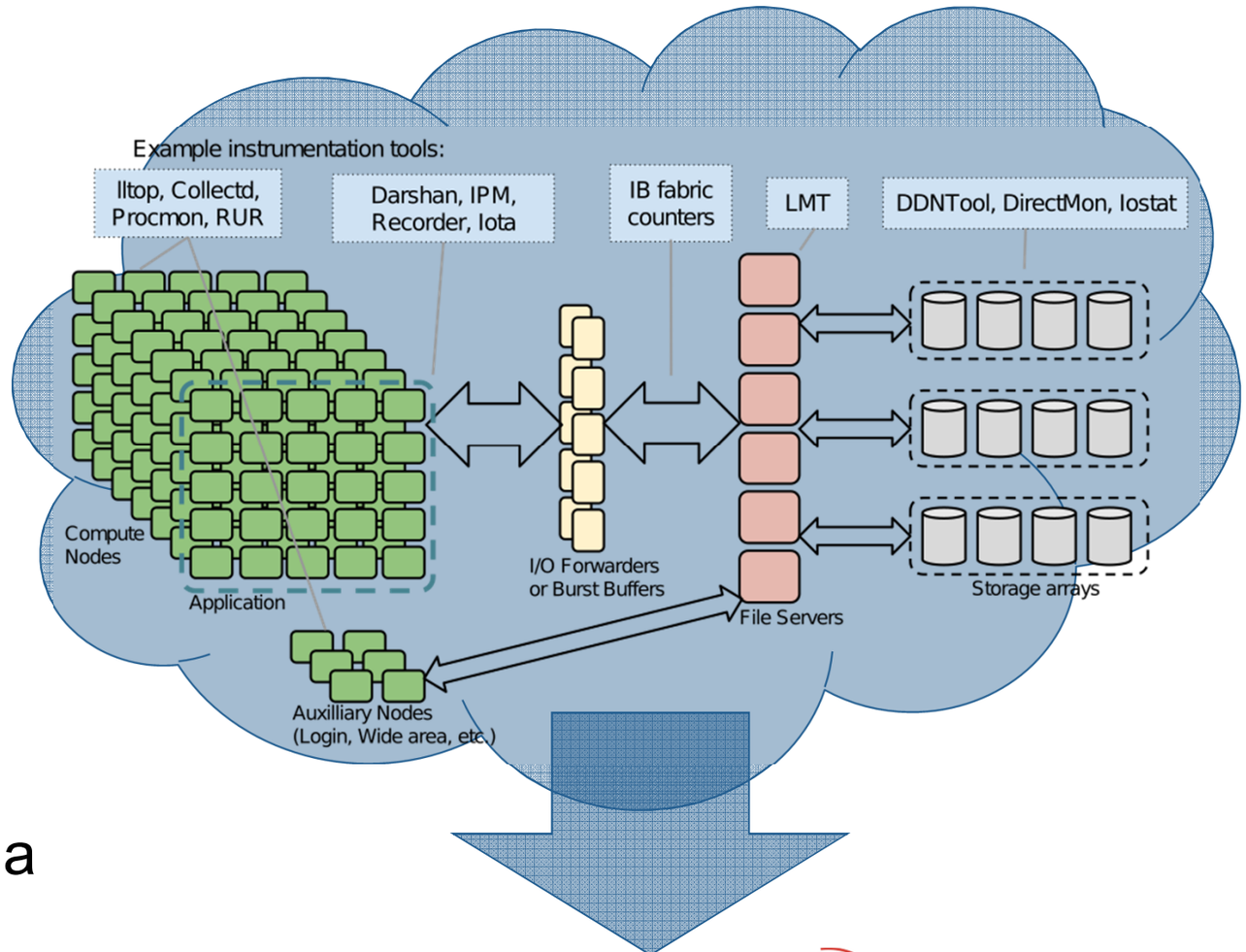
- We need a big picture view
- No lack of instrumentation methods for system components…
  - but with divergent data formats, resolutions, and scope



https://www.nersc.gov/research-and-development/tokio/

Characterizing the I/O system

- 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



TOKIO Strategy

The TOKIO project is a collaboration between LBL and ANL
PI: Nick Wright (LBL), Collaborators: Suren Byna, Glenn Lockwood, William Yoo, Prabhat, Jialin Liu (LBL) Phil Carns, Shane Snyder, Kevin Harms, Zach Nault, Matthieu Dorier, Rob Ross (ANL)

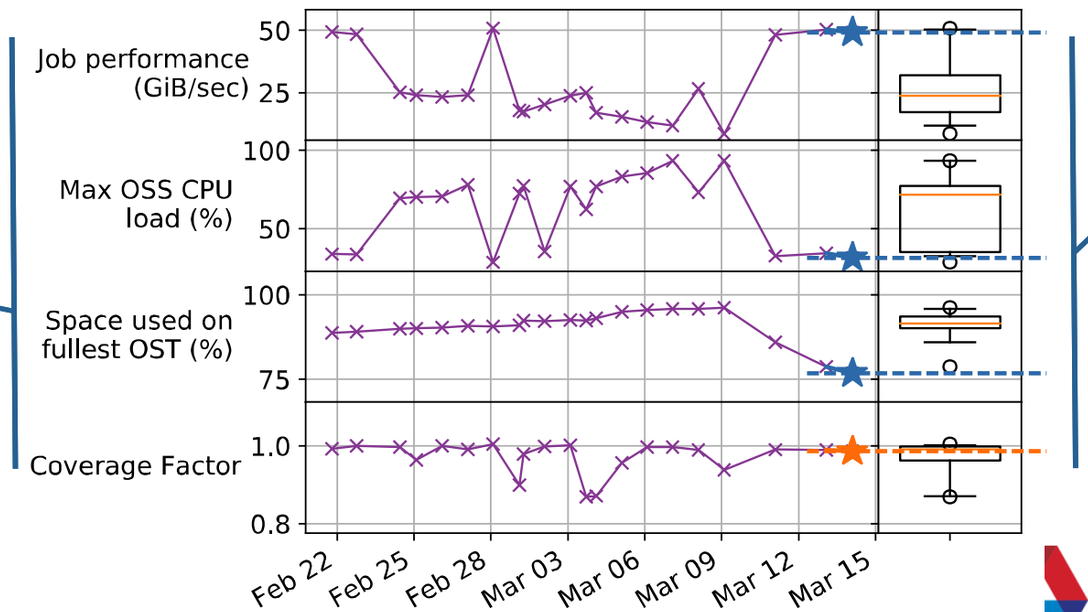
- Integrate existing best-in-class instrumentation tools with help from vendors
- Index and query data sources in their native format
  - Infrastructure to align and link data sets
  - Adapters/parsers to produce coherent views on demand
- Develop integration and analysis methods
- Produce tools that share a common interface and data format
  - Correlation, data mining, dashboards, etc.

UMAMI example

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

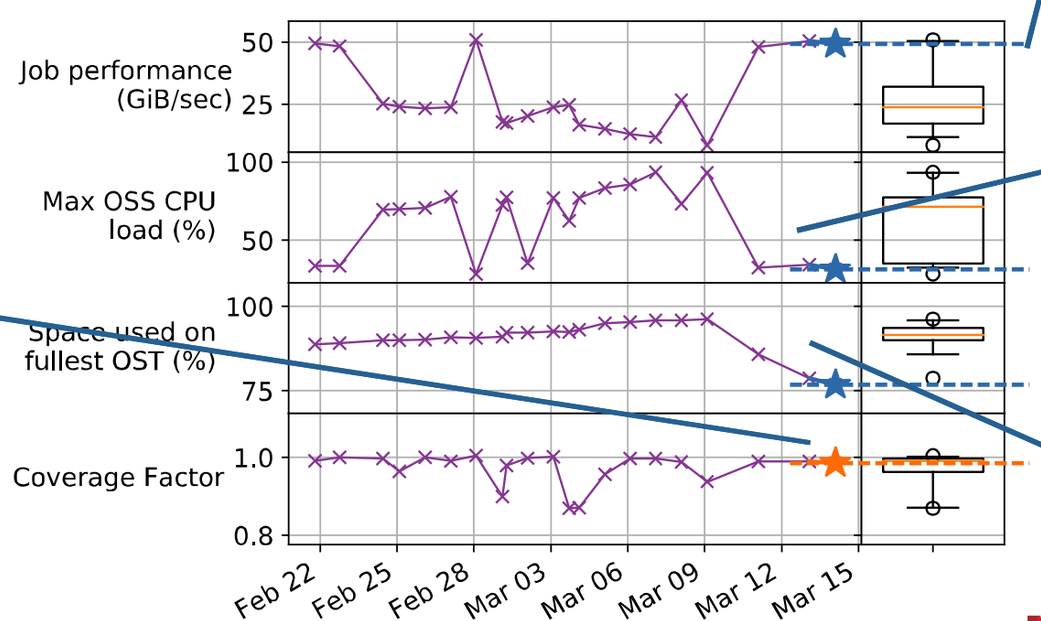
(figures courtesy of Glenn Lockwood, NERSC)

UMAMI example

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

Argonne NATIONAL LABORATORY

ECP EXASCALE COMPUTING PROJECT

Hands on exercises

https://xgitlab.cels.anl.gov/ATPESC-IO/hands-on-2017

- There are hands-on exercises available for you to try out during the day or in tonight's session
  - Demonstrates running applications and analyzing I/O on Theta
  - Try some examples and see if you can find the I/O problem!
- We can also answer questions about your own applications
  - Try it on Theta, Mira, Cetus, Vesta, Cori, Edison, or Titan
  - (note: the Mira, Vesta, and Cetus Darshan versions are a little older and will differ slightly in details from this presentation)

Next up!

- This presentation covered how to evaluate I/O and tune your application.
- The next presentation will walk through the HDF5 data management library.