BERKELEY LAB
Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY
Office of Science

Understanding Parallel I/O Performance and Tuning

Suren Byna

SNTA 2022 Keynote Speech

Contributions from:

- ExaHDF5
- TOKIO - total knowledge of I/O
- Scientific Data Services (SDS)
- ECP ExaIO
- EOD-HDF5

Scope of this presentation

- What is parallel I/O?
- Understanding I/O performance
  - System-wide
  - Application-level
- Tuning I/O performance
- Remaining challenges

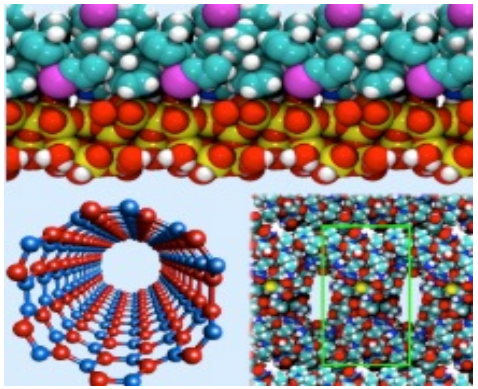
- Papers used in this presentation
  - Jean Luca Bez, Houjun Tang, Bing Xie, David Williams-Young, Rob Latham, Rob Ross, Sarp Oral, and Suren Byna, "I/O Bottleneck Detection and Tuning: Connecting the Dots using Interactive Log Analysis", 6th International Parallel Data Systems Workshop (PDSW) 2021, held in conjunction with SC21
  - Babak Behzad, Suren Byna, Stefan Wild, Prabhat and Marc Snir, "Dynamic Model-driven Parallel I/O Performance Tuning", IEEE Cluster 2015
  - B. Behzad, S. Byna, S. Wild, Prabhat, and M. Snir, "Improving Parallel I/O Autotuning with Performance Modeling", The 23rd ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC) 2014
  - B. Behzad, H. Luu, J. Huchette, S. Byna, Prabhat, R. Aydt, Q. Koziol, and M. Snir, "Taming Parallel I/O Complexity with Auto-Tuning", ACM/IEEE Supercomputing 2013 (SC13)
  - Teng Wang, Suren Byna, Glenn Lockwood, Philip Carns, Shane Snyder, Sunggon Kim, and Nicholas Wright, "A Zoom-in Analysis of I/O Logs to Detect Root Causes of I/O Performance Bottlenecks", IEEE/ACM CCGrid 2019
  - Teng Wang, Suren Byna, Glenn Lockwood, Nicholas Wright, Phil Carns, and Shane Snyder, "IOMiner: Large-scale Analytics Framework for Gaining Knowledge from I/O Logs", IEEE Cluster 2018.
  - Glenn Lockwood, Shane Snyder, Teng Wang, Suren Byna, Phil Carns, and Nicholas Wright, "A Year in the Life of a Parallel File System", 2018 International Conference for High Performance Computing, Networking, and Storage (SC'18)

More related papers from our team …

- Jean Luca Bez, Ahmad Maroof Karimi, Arnab K. Paul, Bing Xie, Suren Byna, Philip Carns, Sarp Oral, Feiyi Wang, and Jesse Hanley, "Access Patterns and Performance Behaviors of Multi-layer Supercomputer I/O Subsystems under, production Load", HPDC 2022
- Jiwoo Bang, Chungyong Kim, Kesheng Wu, Alex Sim, Suren Byna, Hanul Sung, Hyeonsang Eom , "An In-Depth I/O Pattern Analysis in HPC Systems", 28th IEEE International Conference on High Performance Computing, Data, & Analytics (HiPC 2021)
- Bing Xie, Houjun Tang, Suren Byna, Jesse Hanley, Quincey Koziol, Tonglin Li, Sarp Oral, "Battle of the Defaults: Extracting Performance Characteristics of HDF5 under Production Load", CCGrid 2021
- Bing Xie, Houjun Tang, Suren Byna, Quincey Koziol, and Sarp Oral, "Tuning I/O Performance on Summit – HDF5 Write Use Case Study", Invited talk at the HPC I/O in the Data Center Workshop (HPC-IODC) 2020, in conjunction with the ISC High Performance 2020.
- Sunggon Kim, Alex Sim, Kesheng Wu, Suren Byna, Yongseok Son, and Hyeonsang Eom, "Towards HPC I/O Performance Prediction through Large-scale Log Analysis", The 29th International Symposium on High-Performance Parallel and Distributed Computing (HPDC) 2020
- Jiwoo Bang, Chungyong Kim, Kesheng Wu, Alex Sim, Suren Byna, Sunggon Kim, and Hyeonsang Eom, "HPC Workload Characterization Using Feature Selection and Clustering", 3rd International Workshop on System and Network Telemetry and Analytics (SNTA'20), 2020
- Tirthak Patel, Suren Byna, Glenn K. Lockwood, Nicholas J. Wright, Philip Carns, Rob Ross, and Devesh Tiwari, "Uncovering Access, Reuse, and Sharing Characteristics of I/O-Intensive Files on Large-Scale Production HPC Systems", FAST ‘20
- Glenn K. Lockwood, Shane Snyder, Suren Byna, Philip Carns, and Nicholas J. Wright, "Understanding Data Motion in the Modern HPC Data Center", PDSW 2019, in conjunction with SC19.
- Megha Agarwal, Divyansh Singhvi, Preeti Malakar, and Suren Byna, "Active Learning-based Automatic Tuning and Prediction of Parallel I/O Performance", PDSW 2019, in conjunction with SC19
- Tirthak Patel, Suren Byna, Glenn K. Lockwood, and Devesh Tiwari, "Revisiting I/O Behavior in Large-Scale Storage Systems: The Expected and the Unexpected", SC19
- Glenn Lockwood, Shane Snyder, et al., "UMAMI: A Recipe for Generating Meaningful Metrics through Holistic I/O Performance Analysis", 2nd PDSW-DISCS, 2017 (Held in conjunction with SC17)
- Cong Xu, Shane Snyder, Omkar Kulkarni, Vishwanath Venkatesan, Philip Carns, Suren Byna, Robert Sisneros, and Kalyana Chadalavada, "DXT: Darshan eXtended Tracing", Cray User Group Conference 2017 (CUG 2017)
- Cong Xu, Suren Byna, Vishwanath Venkatesan, Robert Sisneros, Omkar Kulkarni, Mohamad Chaarawi, and Kalyana Chadalavada, "LIOProf: Exposing Lustre File System Behavior for I/O Middleware", CUG 2016
- Babak Behzad, Suren Byna, Prabhat and Marc Snir, "Pattern-driven Parallel I/O Tuning ", 10th Parallel Data Storage Workshop (PDSW) 2015, in conjunction with SC15, November 2015
- H. Luu, M. Winslett, W. Gropp, R. Ross, P. Carns, K. Harms, Prabhat, S. Byna, and Y. Yao, "A Multi-platform Study of I/O Behavior on Petascale Supercomputers", The 24th ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC) 2015

Fast I/O of data is critical for science on HPC systems

- Scientific simulations, experiments, and observations are producing terabytes to petabytes of data



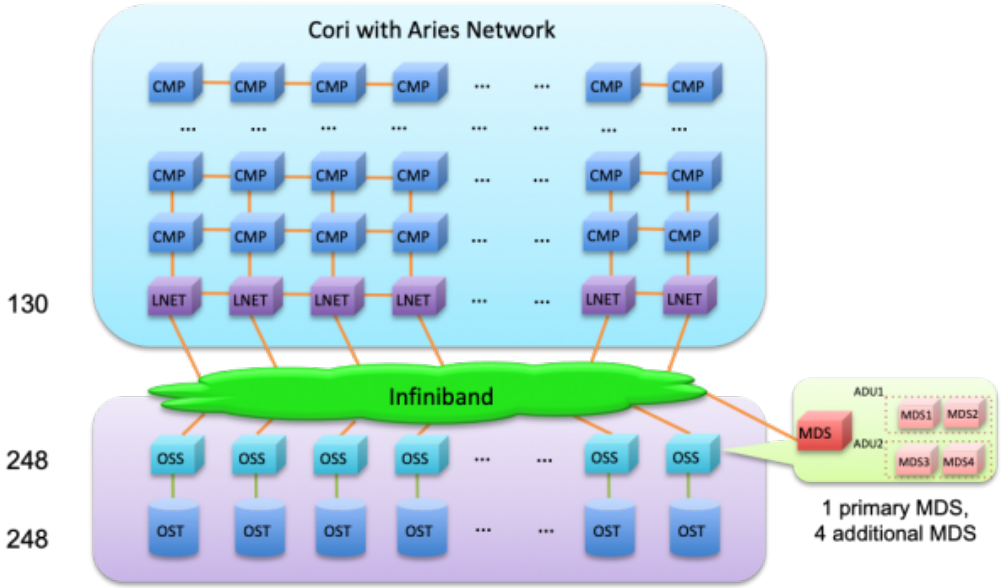
Chemistry
(1.3PB/job)

Molecular
(818TB/job)

Cosmology
(462TB/job)

- Understanding and optimizing data movement performance is crucial in high performance computing

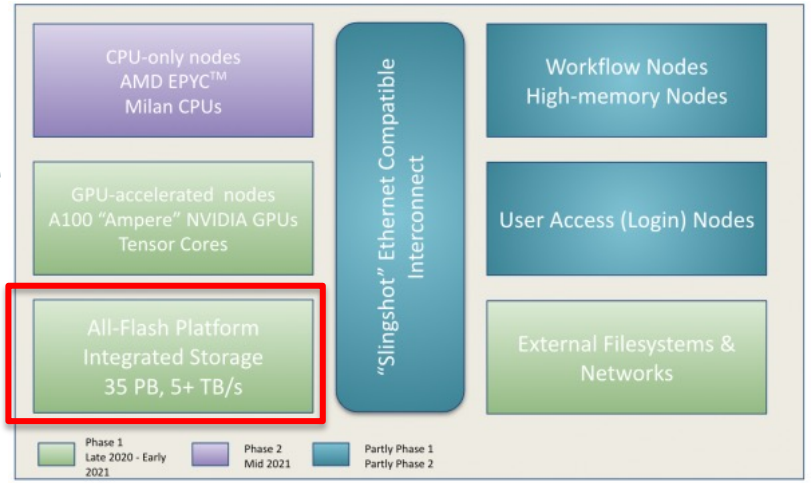
HPC Storage architectures are diverse

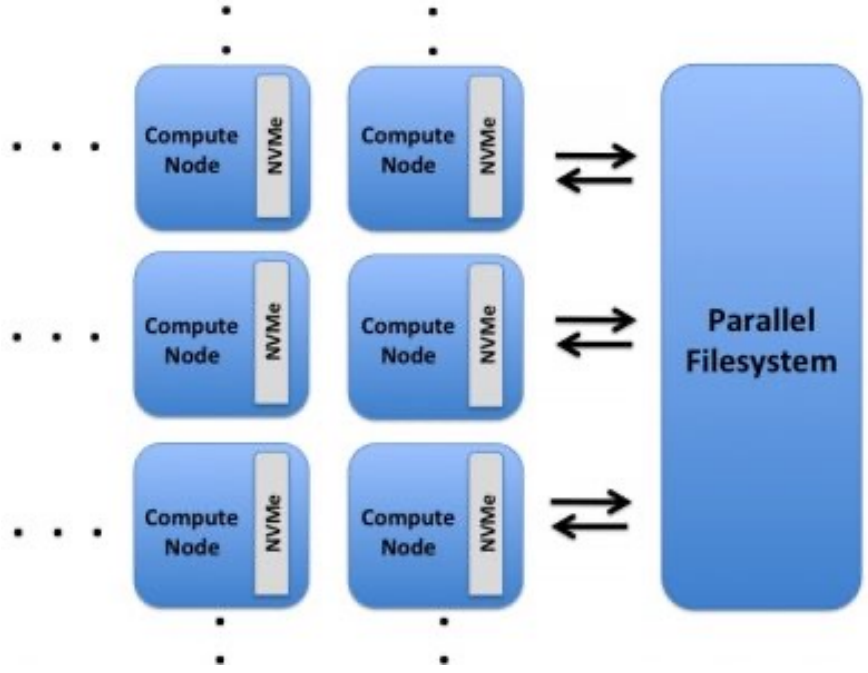


Each OSS controls one OST. The Infiniband connects the MDS, ADUs and OSSs to the LNET routers on the Cray XC System. The OSTs are configured with GridRAID, similar to RAID6, (8+2), but can restore failure 3.5 times faster than traditional RAID6. Each OST consists of 41 disks, and can deliver 240TB capacity.

Cori's storage

Perlmutter storage

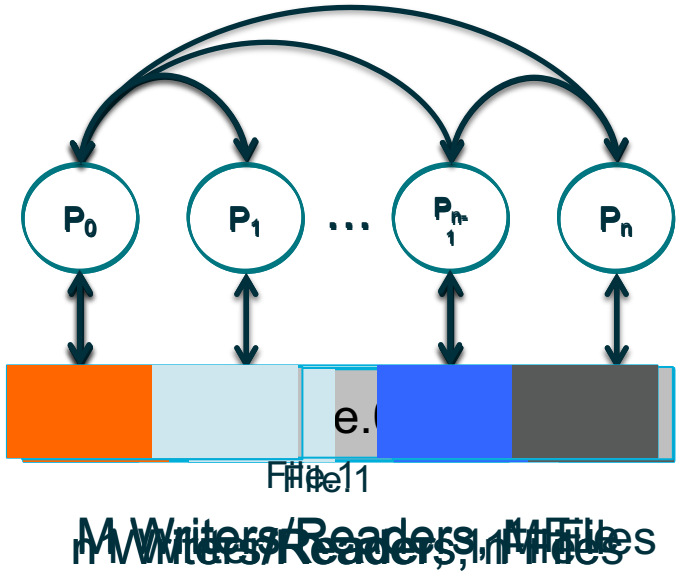




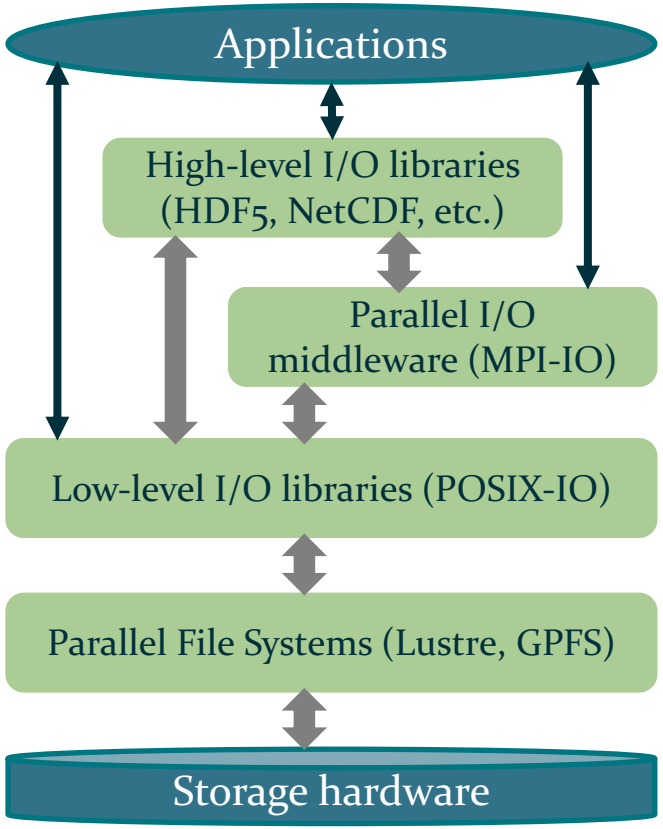
Summit's storage

What is parallel I/O

- **Types of parallel I/O**
  - 1 writer/reader, 1 file
  - N writers/readers, N files (File-per-process)
  - N writers/readers, 1 file
  - M writers/readers, 1 file
    - Aggregators
    - Two-phase I/O
  - M writers/readers, M files (file-per-aggregator)

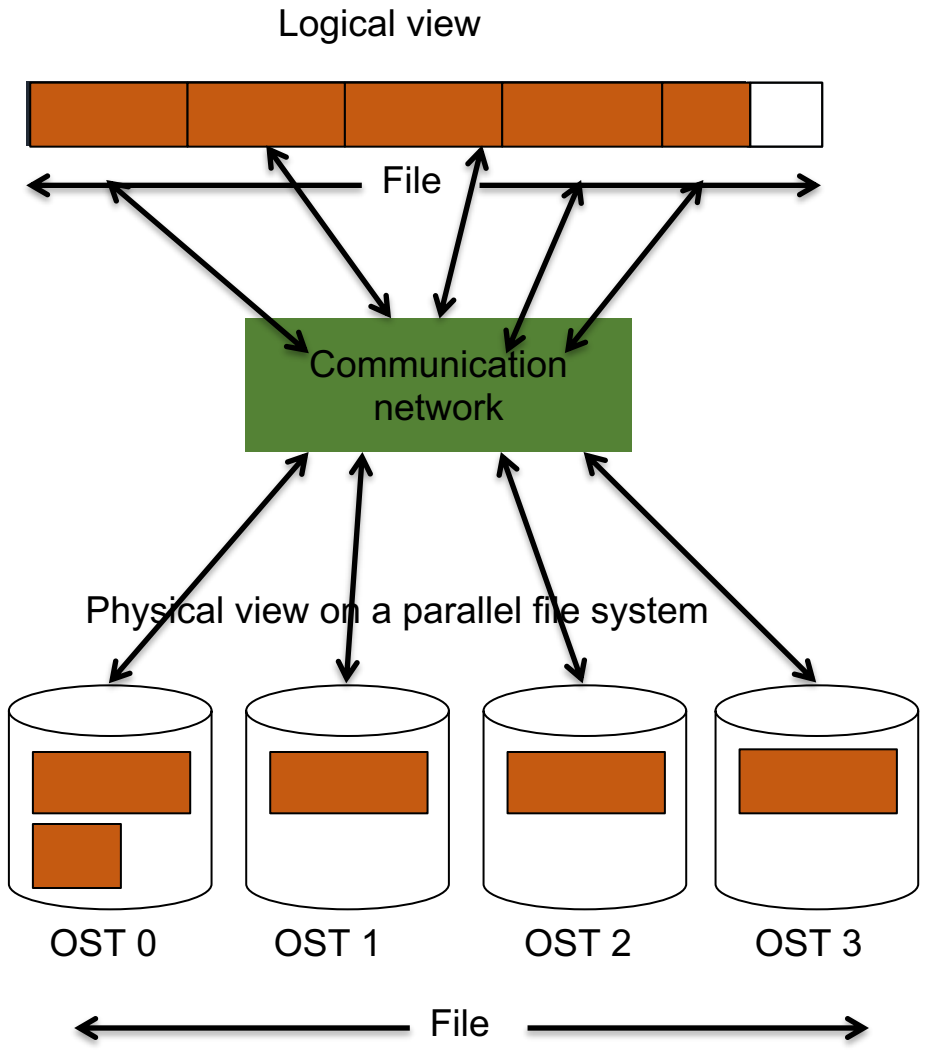


I/O software stack - Several layers with inter-dependencies



I/O software stack - Several layers with inter-dependencies

- Parallel file systems
  - Lustre and Spectrum Scale (GPFS)
- Typical building blocks of parallel file systems
  - Storage hardware – HDD or SSD RAID
  - Storage servers (in Lustre, Object Storage Servers [OSS], and object storage targets [OST]
  - Metadata servers
  - Client-side processes and interfaces
- Management
  - Stripe files for parallelism
  - Tolerate failures



Why is my I/O slow?

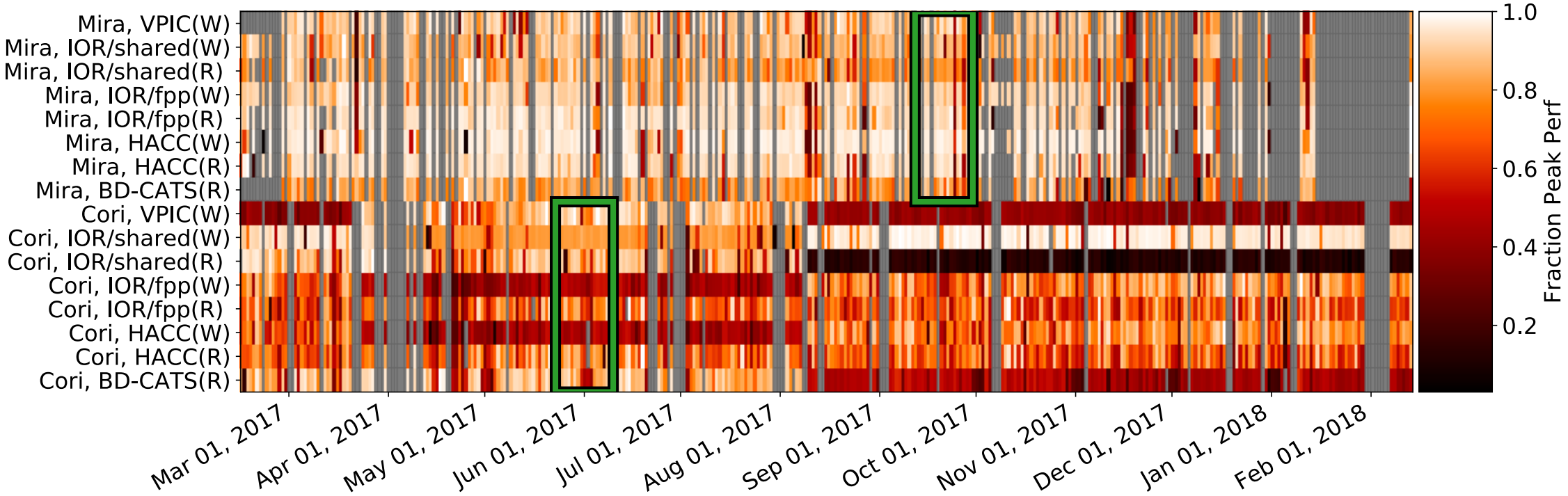
- Another application is interfering with your application
- File system is doing something wrong
- Your application is doing something wrong

Why is my I/O slow?

- Another application is interfering with your application
- File system is doing something wrong
- Your application is doing something wrong

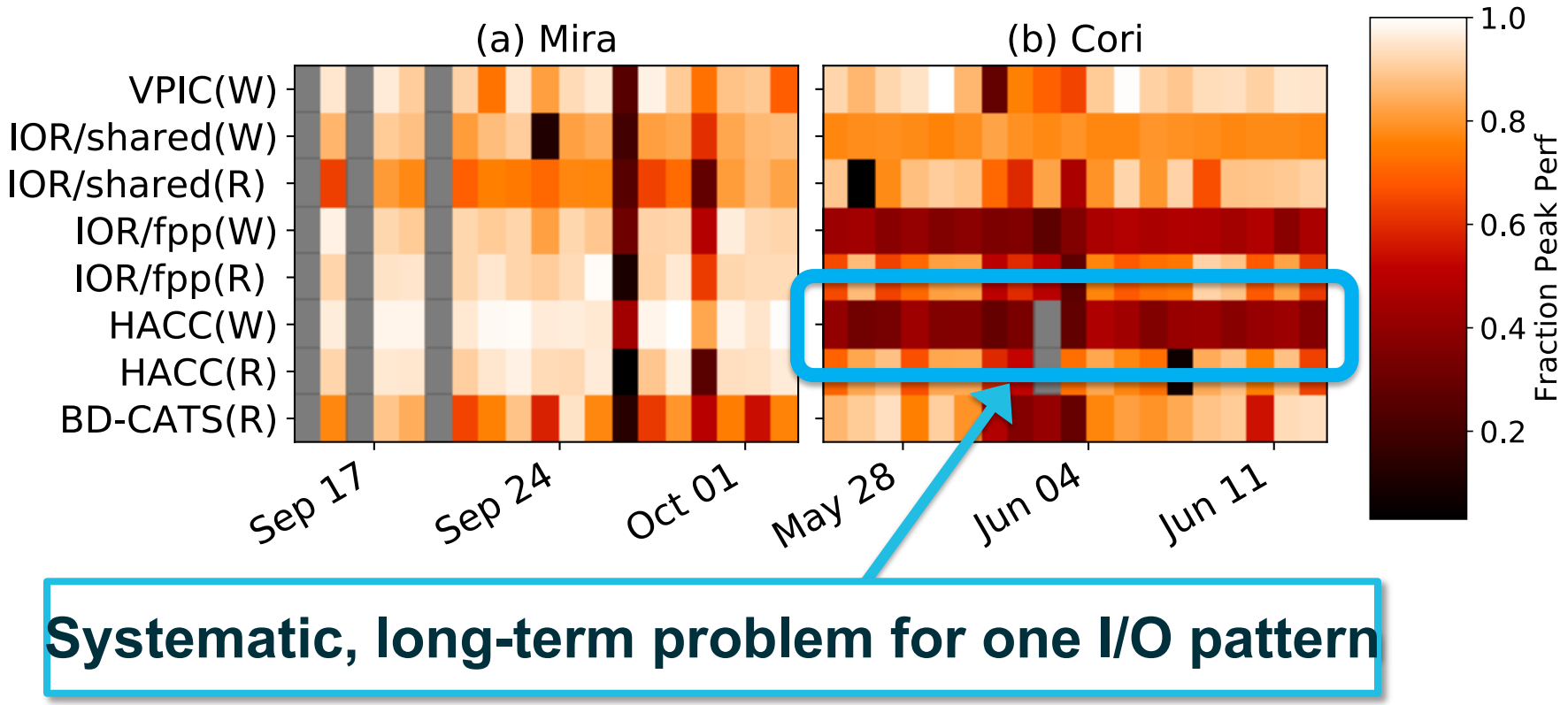
I/O performance variation in production

- Year-long study of same I/O kernels running on two systems
  - Mira at Argonne (GPFS) and Cori at NERSC (Lustre)



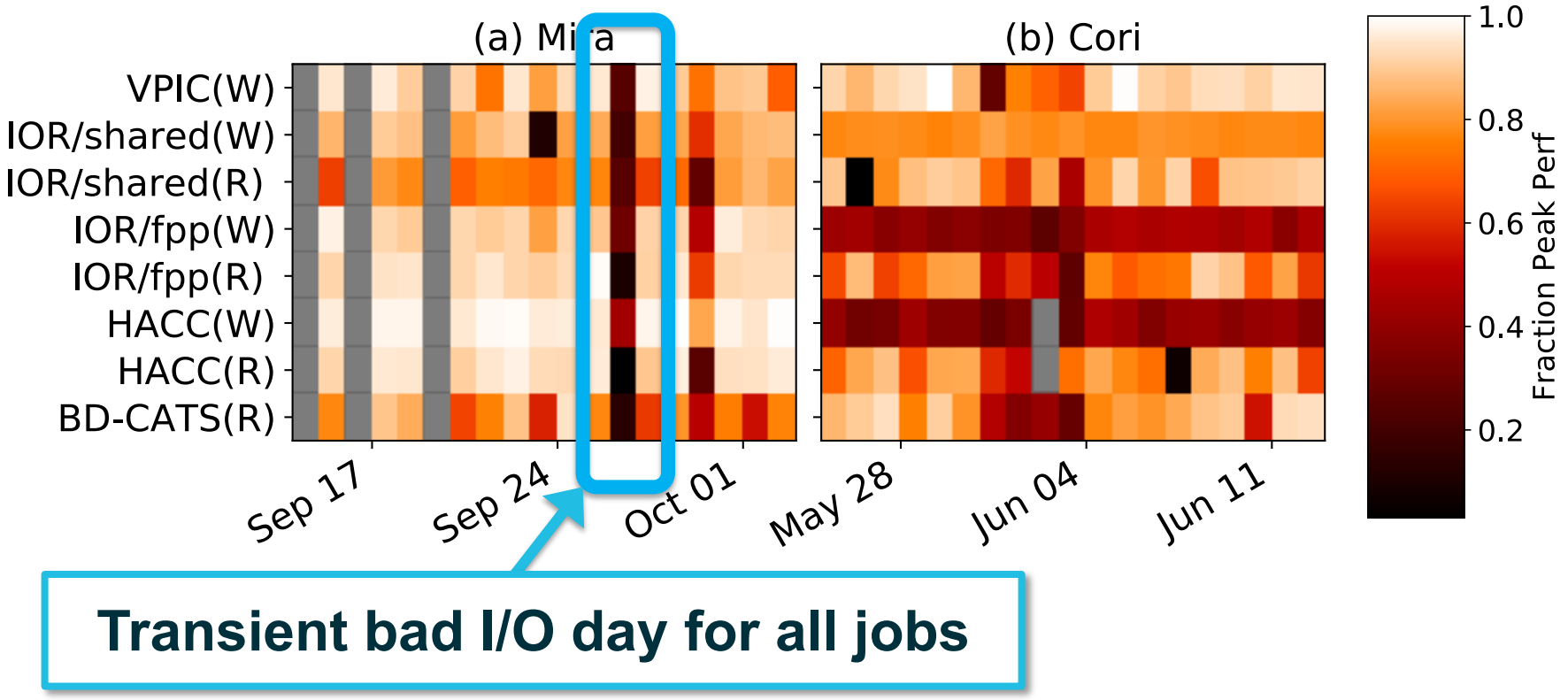
Glenn Lockwood, Shane Snyder, Teng Wang, Suren Byna, Phil Carns, and Nicholas Wright, "A Year in the Life of a Parallel File System", 2018 International Conference for High Performance Computing, Networking, and Storage (SC'18)

Performance varies over the long term



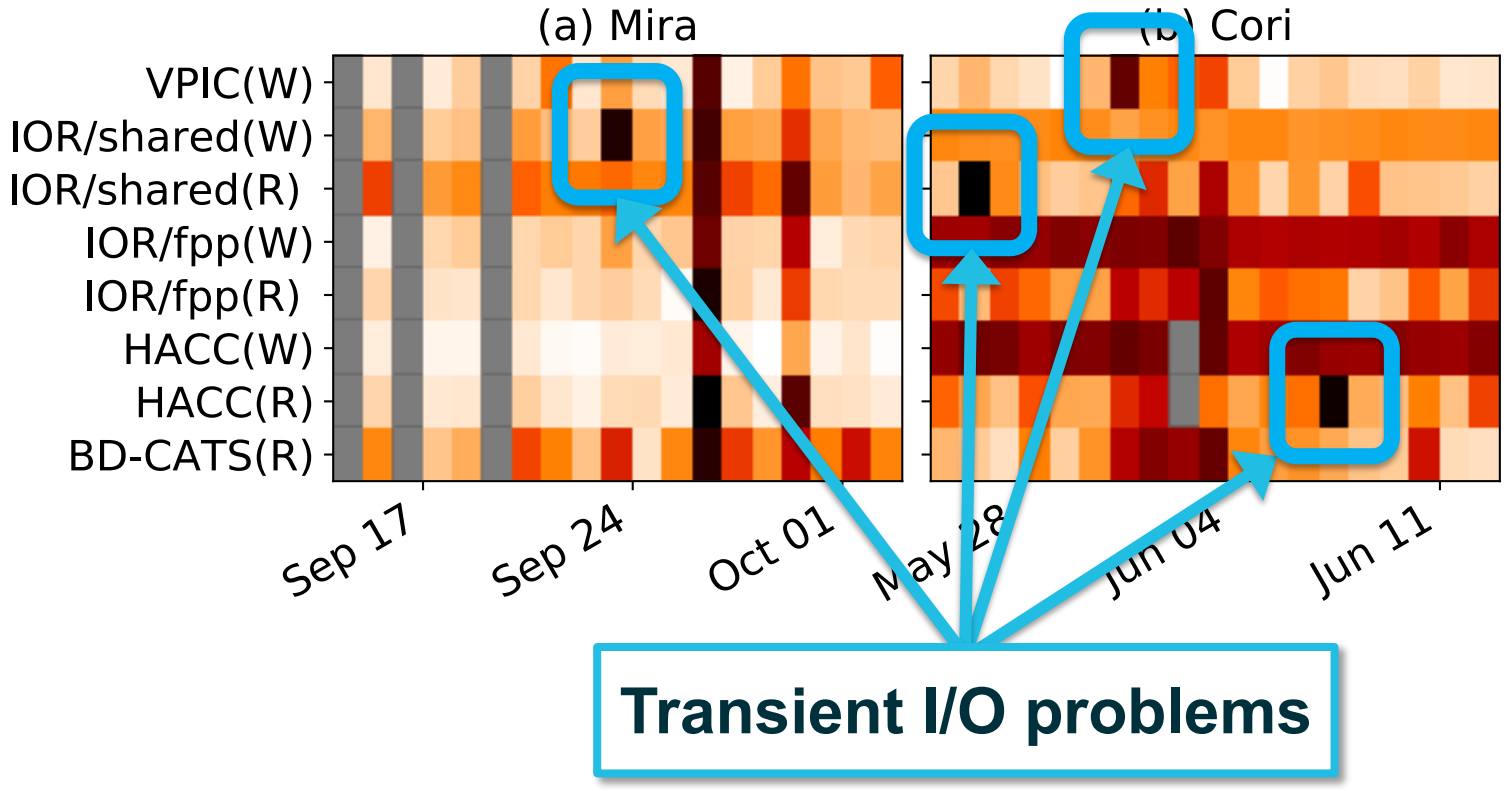
Glenn Lockwood, Shane Snyder, Teng Wang, Suren Byna, Phil Carns, and Nicholas Wright, "A Year in the Life of a Parallel File System", 2018 International Conference for High Performance Computing, Networking, and Storage (SC'18)

Performance varies over the short term



Glenn Lockwood, Shane Snyder, Teng Wang, Suren Byna, Phil Carns, and Nicholas Wright, "A Year in the Life of a Parallel File System", 2018 International Conference for High Performance Computing, Networking, and Storage (SC'18)

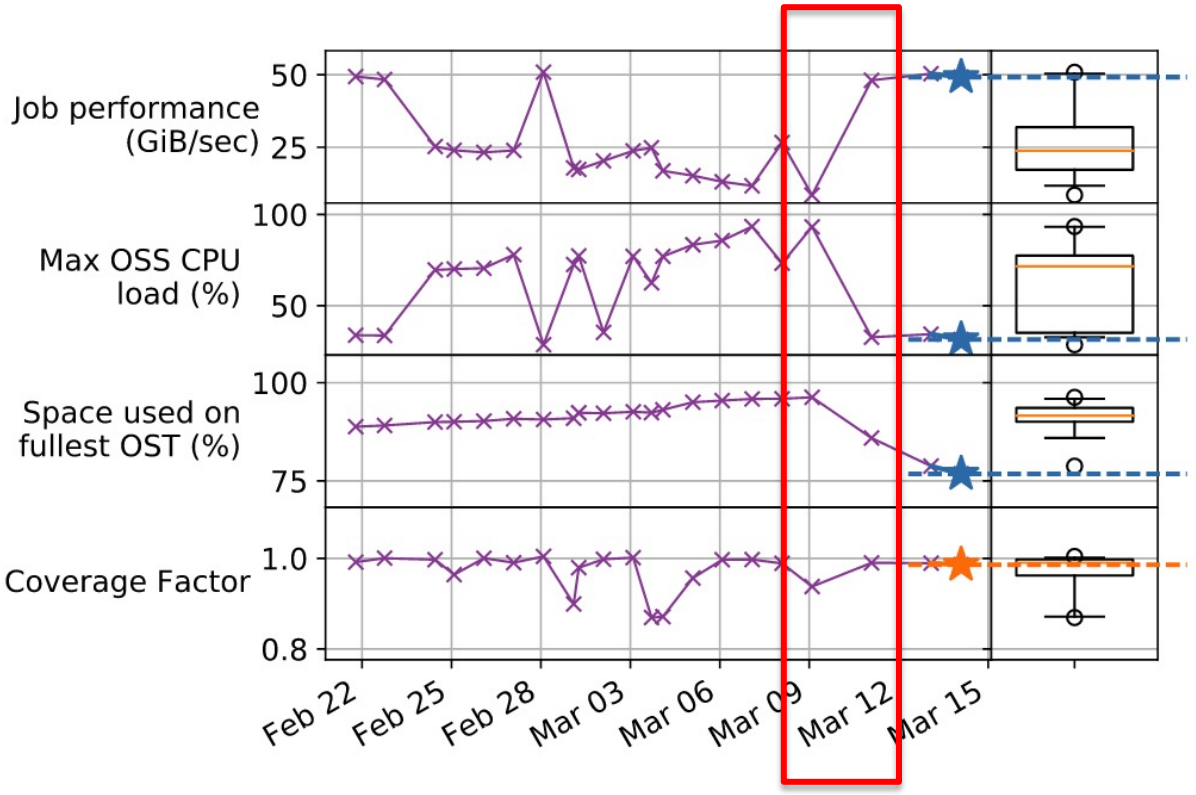
Performance also experiences transient losses



Glenn Lockwood, Shane Snyder, Teng Wang, Suren Byna, Phil Carns, and Nicholas Wright, "A Year in the Life of a Parallel File System", 2018 International Conference for High Performance Computing, Networking, and Storage (SC'18)

Umami: meaningful metrics for understanding I/O load

- Metrics based on Darshan and file system logs
  - Job performance
  - Coverage factors - how's my job doing on the system?
  - Historical measurements of the same application
  - Plots to show variance of current values



https://github.com/nersc/pytokio

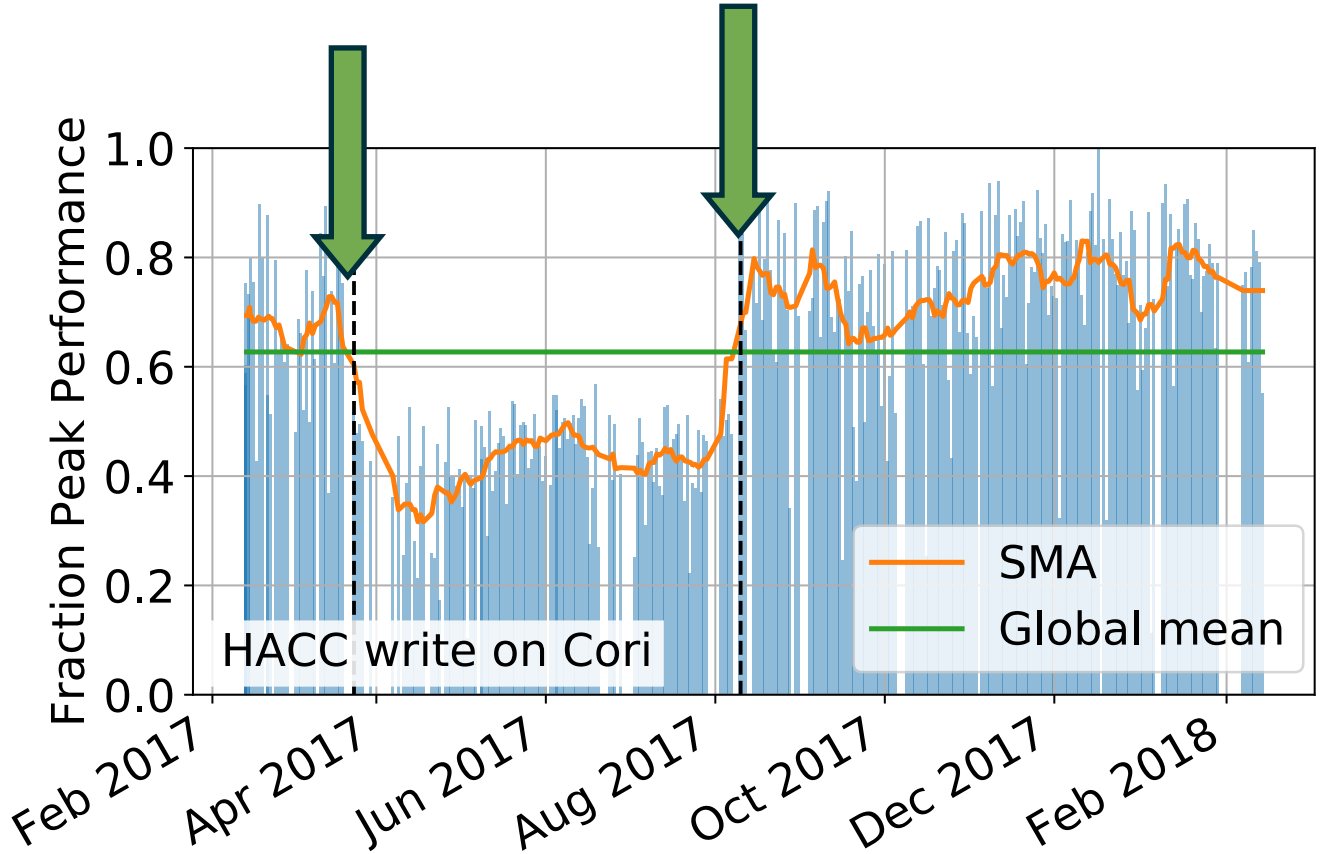
Glenn Lockwood, Shane Snyder, Wucherl Yoo, Kevin Harms, Zachary Nault, Suren Byna, Philip Carns, Nicholas Wright, "UMAMI: A Recipe for Generating Meaningful Metrics through Holistic I/O Performance Analysis", 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems (PDSW-DISCS), 2017 (Held in conjunction with SC17)

Why is my I/O slow?

- Another application is interfering with your application
- File system is doing something wrong
- Your application is doing something wrong

Quantitatively bound long-term problems

- Goal: Numerically distinguish time-dependent variation
- Simple moving averages (SMAs) from financial market technical analysis
- Where short-term average performance diverges from overall average
- Example: Bug in a specific file system client version



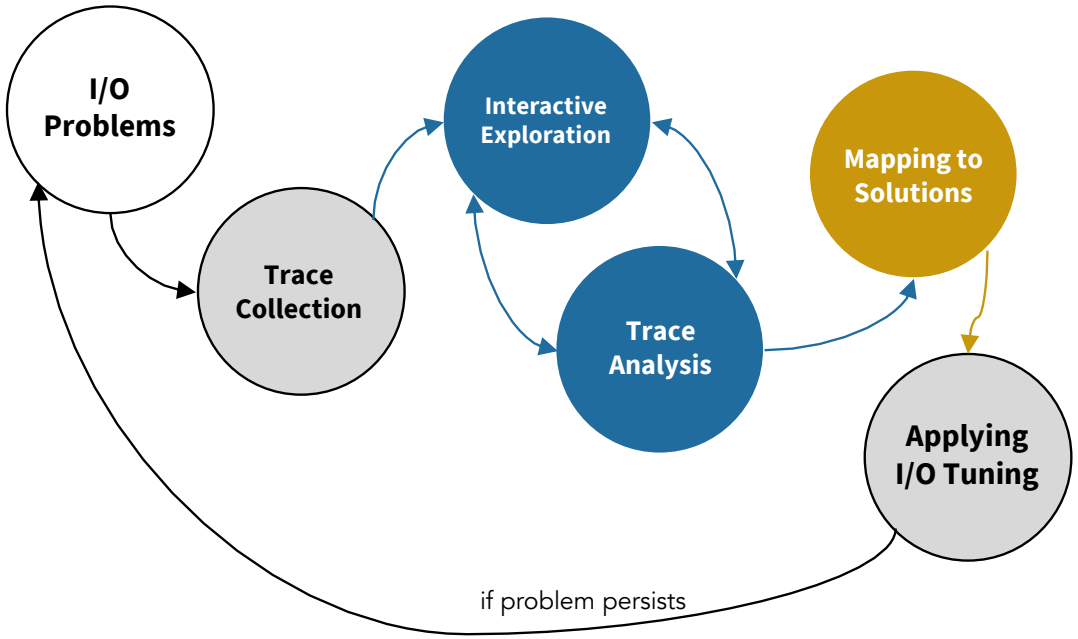
Glenn Lockwood, Shane Snyder, Teng Wang, Suren Byna, Phil Carns, and Nicholas Wright, "A Year in the Life of a Parallel File System", 2018 International Conference for High Performance Computing, Networking, and Storage (SC'18)

Why is my I/O slow?

- Another application is interfering with your application
- File system is doing something wrong
- Your application is doing something wrong

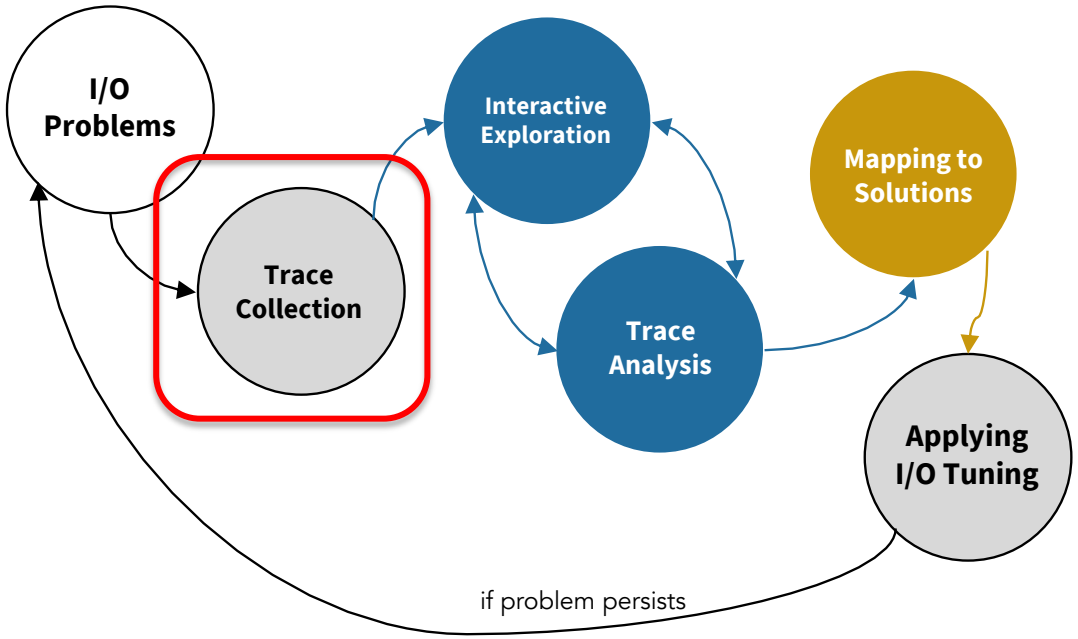
Process to find and solve I/O problems

- Collect logs / traces
- Analyze for finding problems
- Find tuning options
- Apply tuning solutions

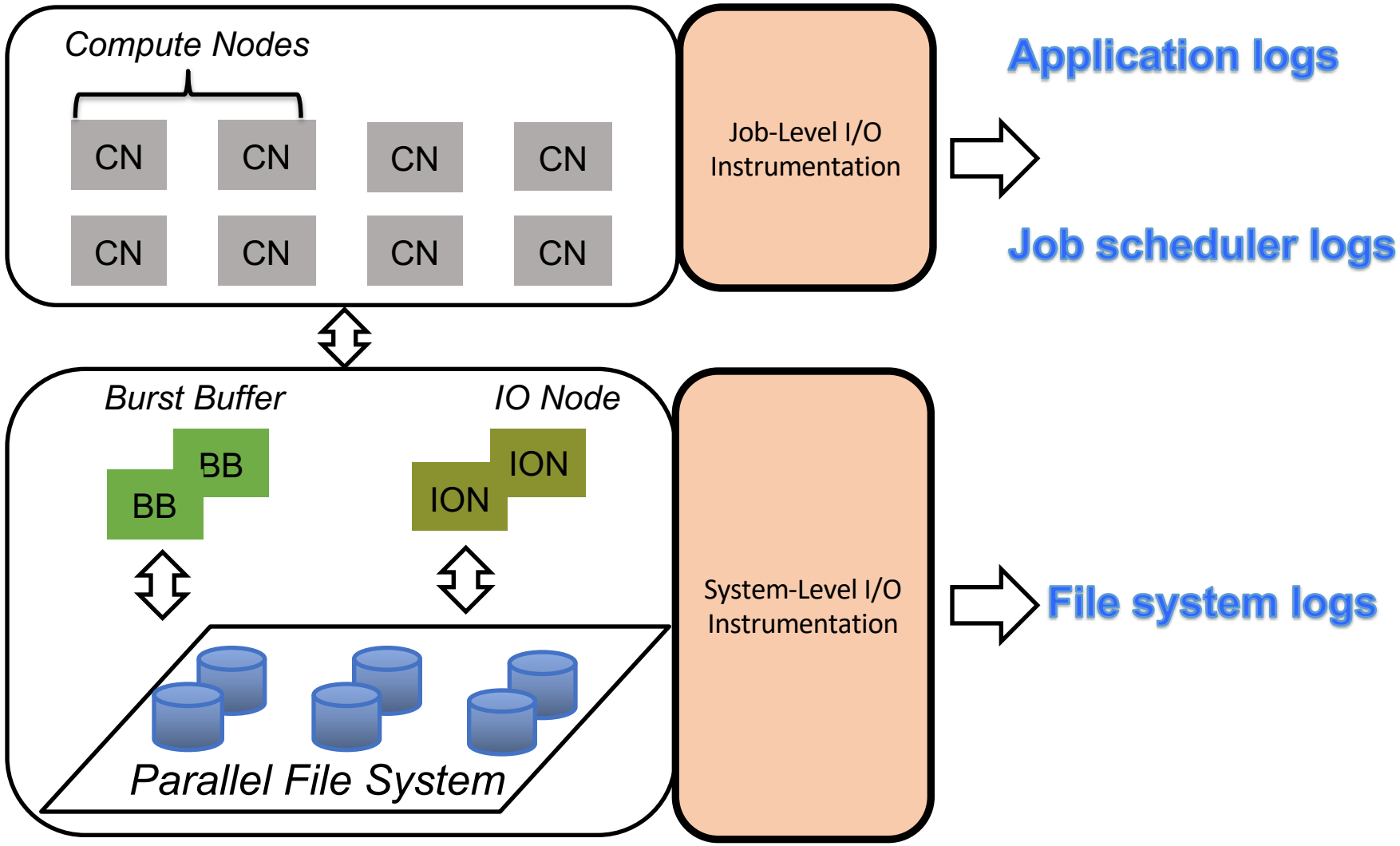


Process to find and solve I/O problems

- Collect logs / traces
- Analyze for finding problems
- Find tuning options
- Apply tuning solutions



Logs at different layers



Logs at different layers

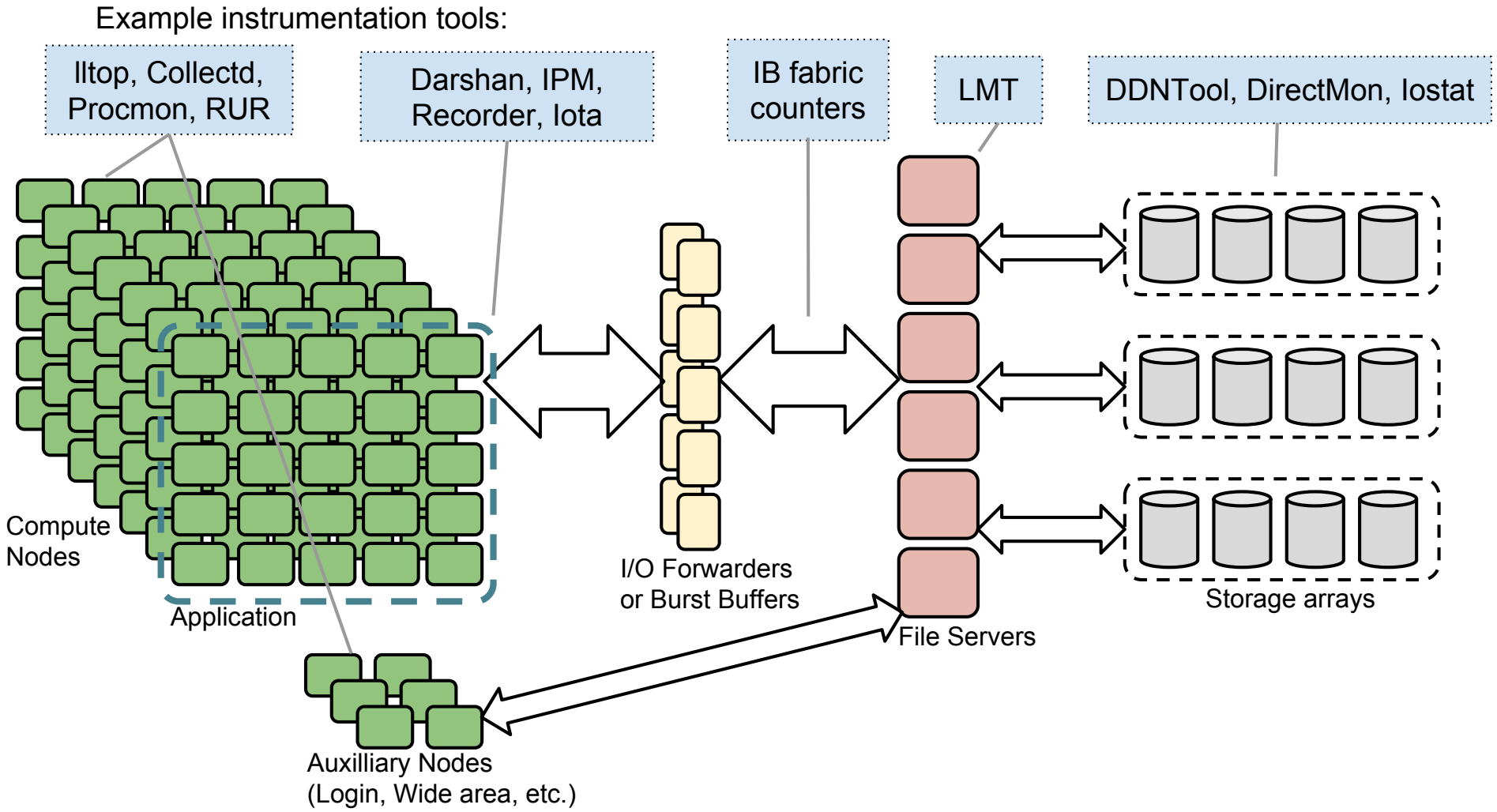


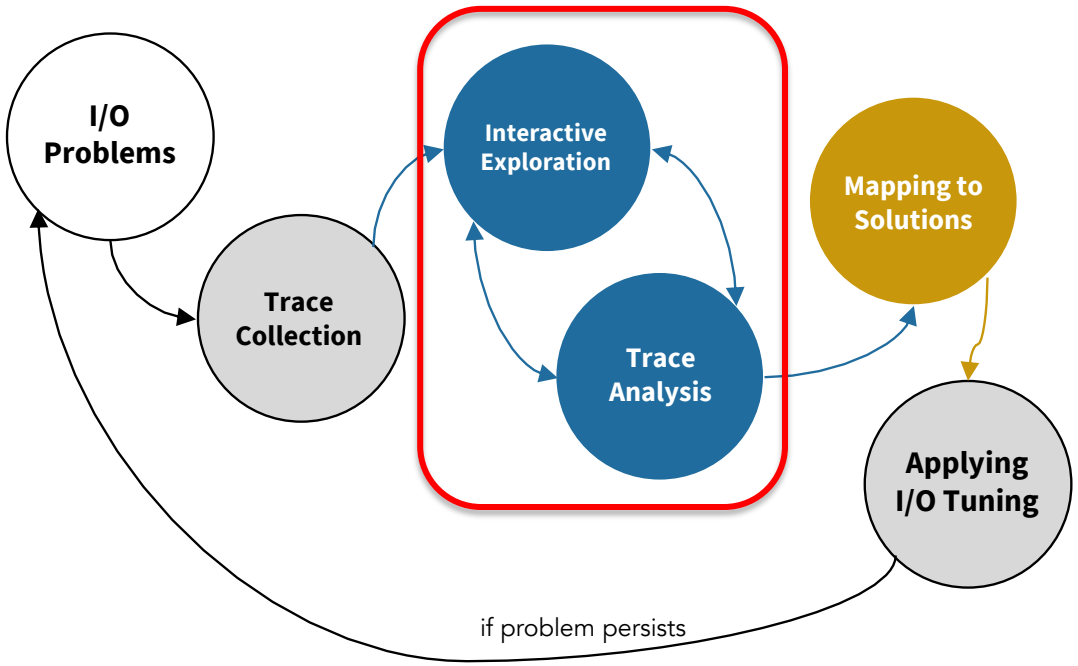
Image from Phil Carns, Argonne National Lab

What's in those logs?

- Analytics based on application logs (Darshan)
  - Job-level I/O statistics: read/write size, read/write time, process count, IO bandwidth, and other metrics on each job and each file.
- Analytics based on scheduler logs (Slurm)
  - Job-level I/O statistics: Finding the number of nodes used by a job.
- Analytics based on file system logs (Lustre – LMT)
  - System-level I/O statistics: Calculating the amount of data written to/read from on the parallel file system during a given period, etc.

Process to find and solve I/O problems

- Collect logs / traces
- Analyze for finding problems
- Find tuning options
- Apply tuning solutions



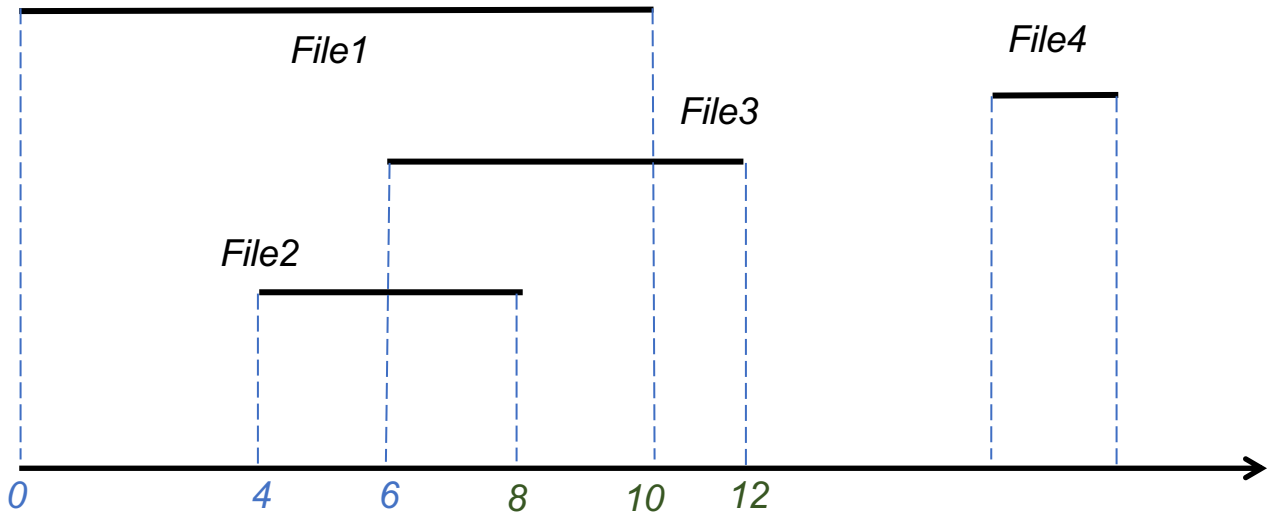
We have logs, then what?

- Understand the timeline

  - When did I/O occur
    - File open / close
    - I/O requests
    - File size, request size, and various characteristics
    - Which storage servers / targets?

Sweep-line analysis

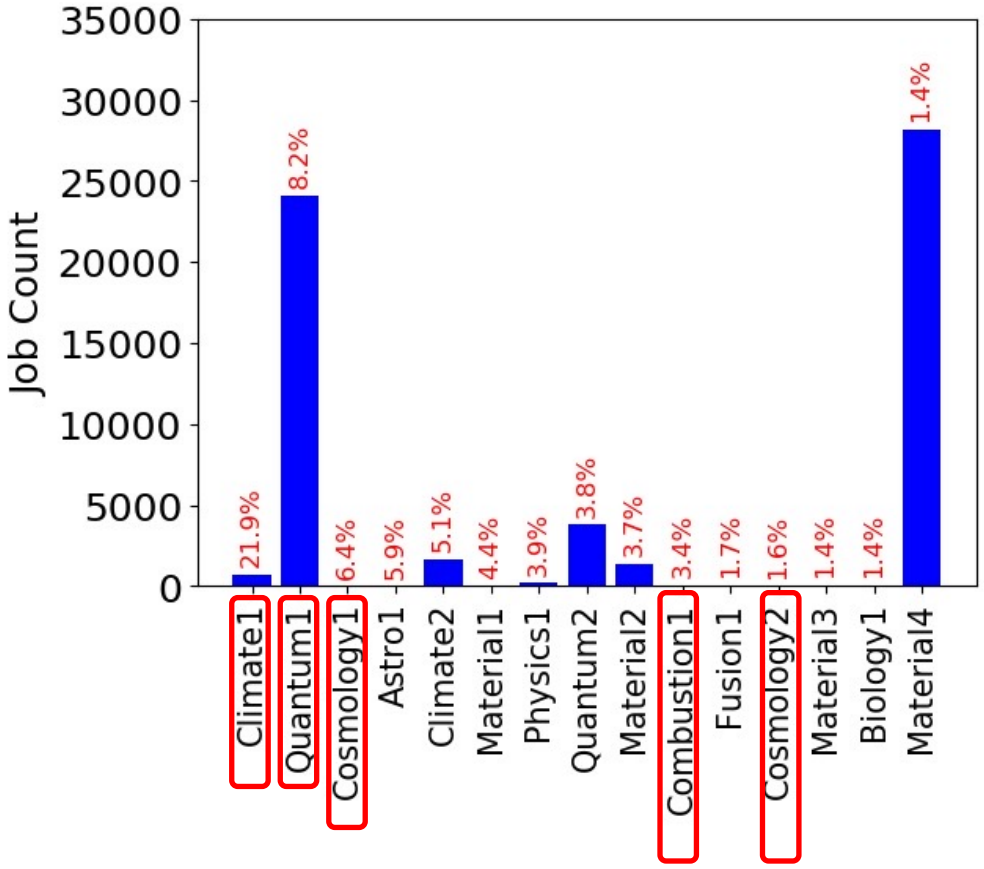
- Sweep-line analysis to extract files on the I/O covering set and focus on analyzing these files
- I/O covering set: the minimum set of files whose I/O times cover the whole job



Teng Wang, Suren Byna, Glenn Lockwood, Philip Carns, Shane Snyder, Sunggon Kim, and Nicholas Wright, "A Zoom-in Analysis of I/O Logs to Detect Root Causes of I/O Performance Bottlenecks", IEEE/ACM CCGrid 2019

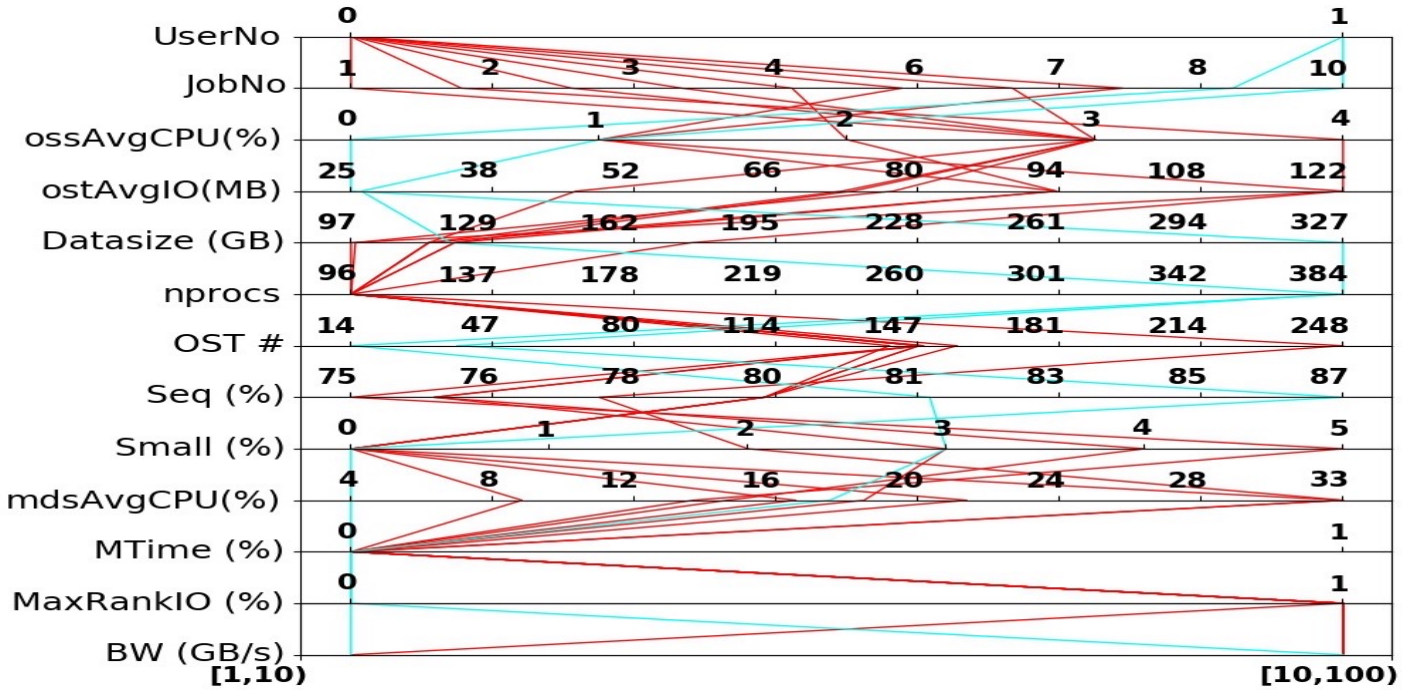
Darshan log analysis

- 88,000 Darshan logs gathered during two-month period.
  - read/write < 1GB and run < 5 minutes
- Selected the top 15 applications with high node hour consumption



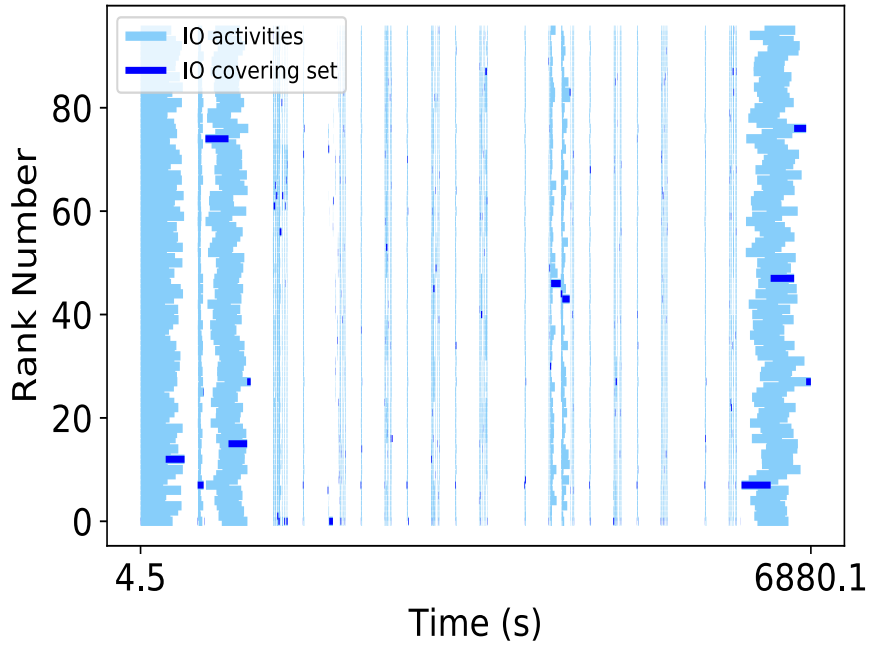
Application-level analysis with parallel coordinates

- Cosmology1's IO is well-formed: all jobs have high sequential IO (>75%), low small IO ratio (< 5%). Low metadata and storage server CPU utilization (<4% and <33%), etc.
  - However, IO bandwidth varies between [1,10) GB/s and [10,100)GB/s, which needs a job-level analysis.

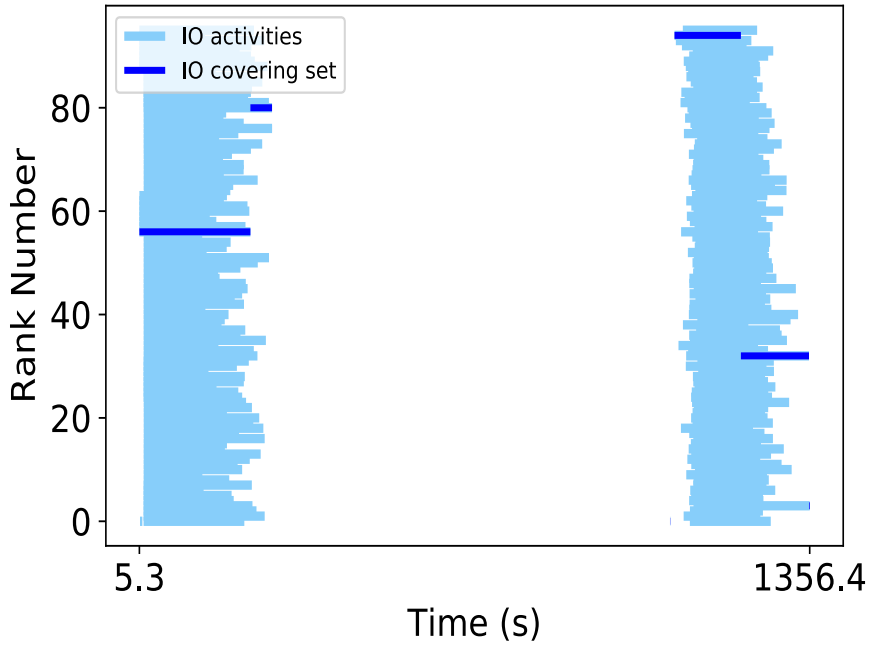


Job-level analysis

- Jobs with [1,10) GB/s is due to its frequent I/O phases, I/O time of each I/O phase is bottlenecked by the slowest rank (a rank is a process).



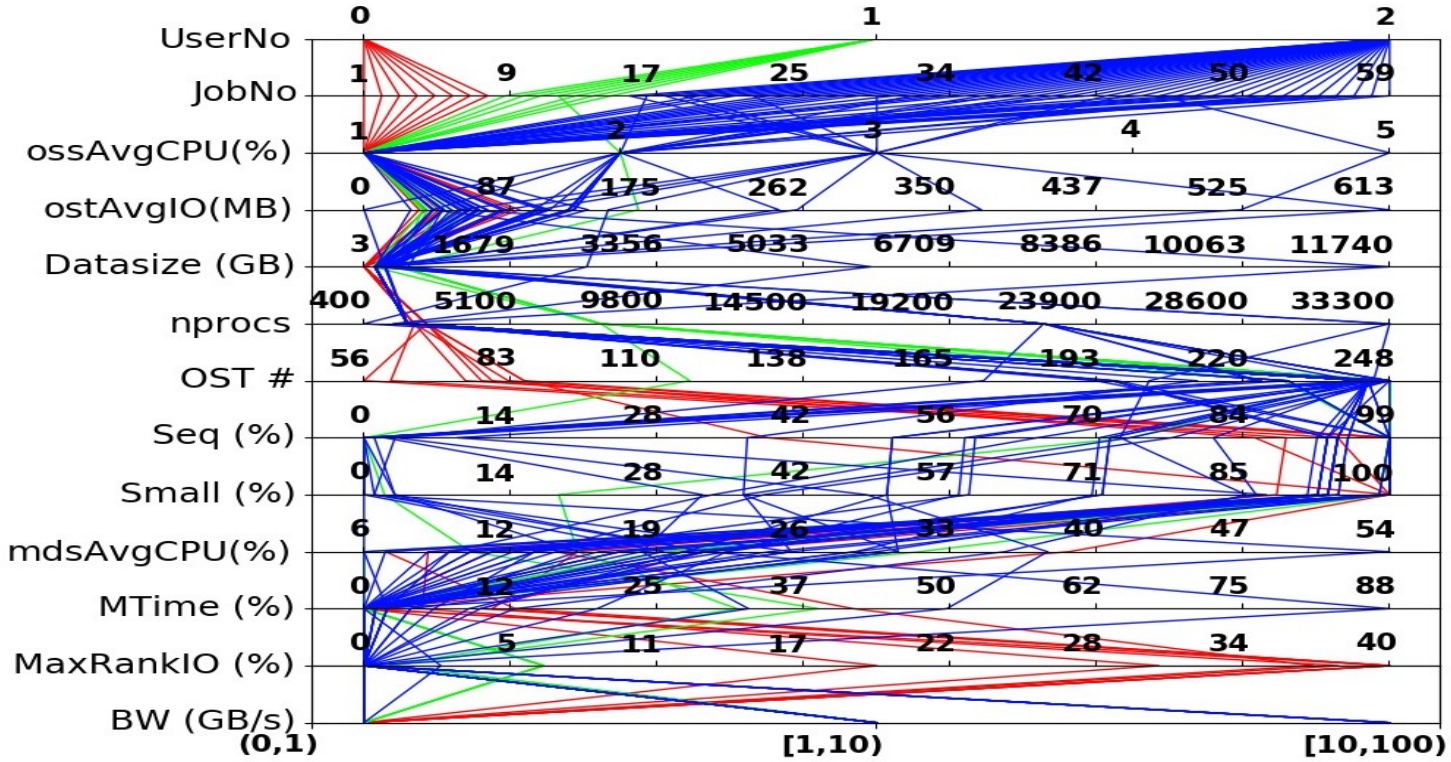
(a) Job with [1, 10) GB/s



(b) Job with [10, 100) GB/s

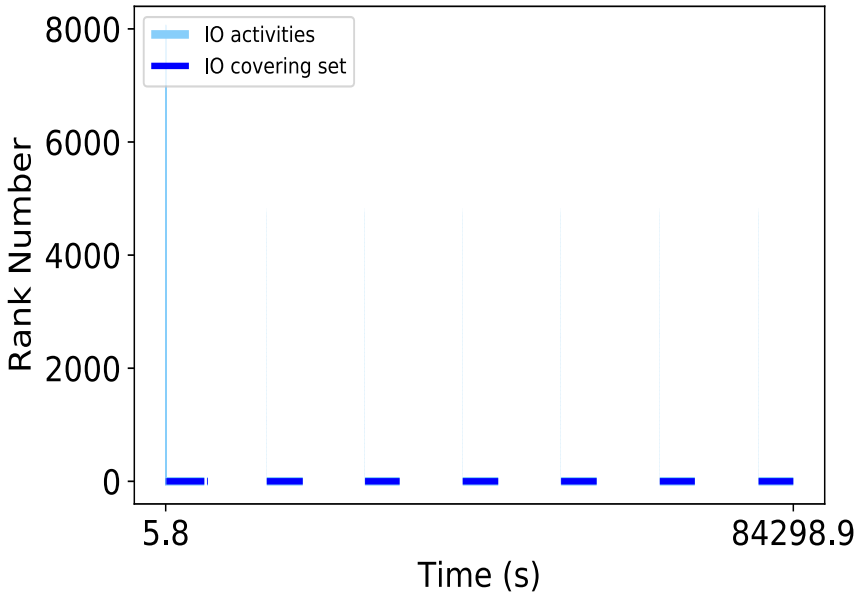
Application-level analysis - Combustion app

- There are 59 jobs belonging to 3 users.
  - User 0's jobs (red) are all bottlenecked by too many small I/O (100%).
  - User 1's jobs (green) fall in both [0, 1) GB/s and [1,10) GB/s.

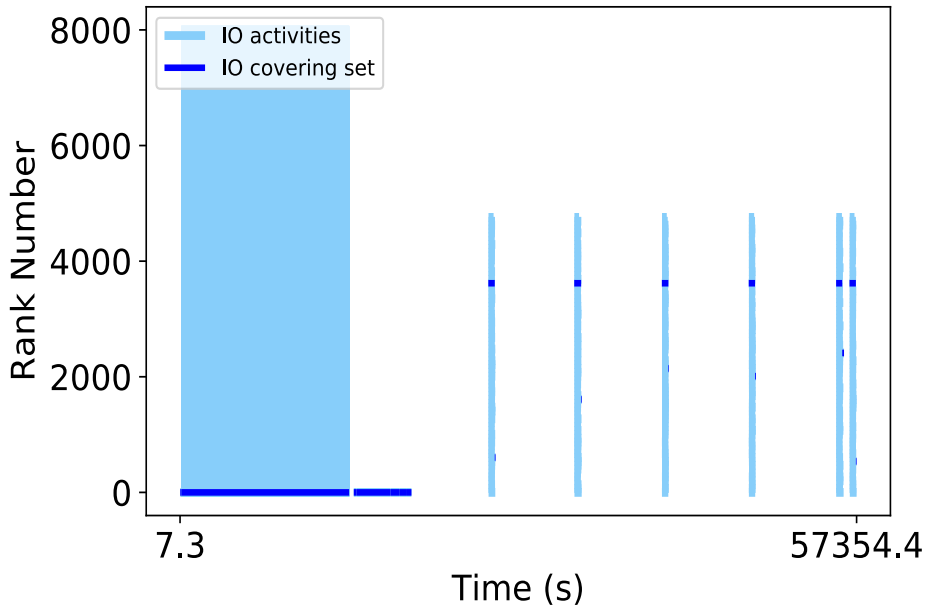


Job-level analysis - Combustion app

- Zoom-in one representative User 1's job in each bandwidth category.
  - [0, 1)GB/s job is bottlenecked by rank 0 undertaking excessively higher IO workload.
  - [1, 10) GB/s jobs is bottlenecked by all ranks reading a shared file, but Darshan log indicates only one rank performs actual read



(a) Job with [0, 1) GB/s



(b) Job with [1, 10) GB/s

Teng Wang, Suren Byna, Glenn Lockwood, Philip Carns, Shane Snyder, Sunggon Kim, and Nicholas Wright, "A Zoom-in Analysis of I/O Logs to Detect Root Causes of I/O Performance Bottlenecks", IEEE/ACM CCGrid 2019

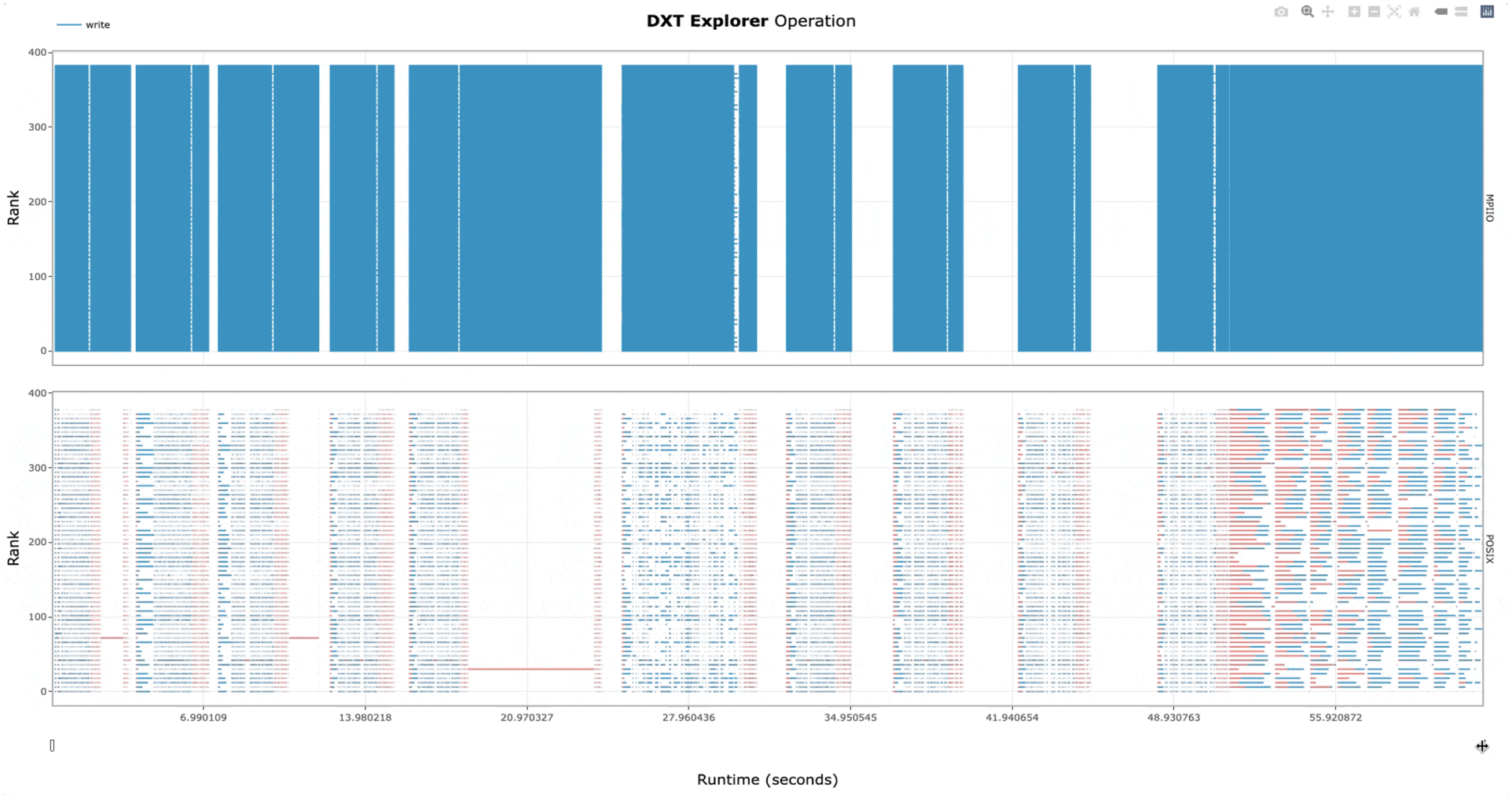
Interactive visualization of I/O timeline

github.com/hpc-io/dxt-explorer

docker pull hpcio/dxt-explorer

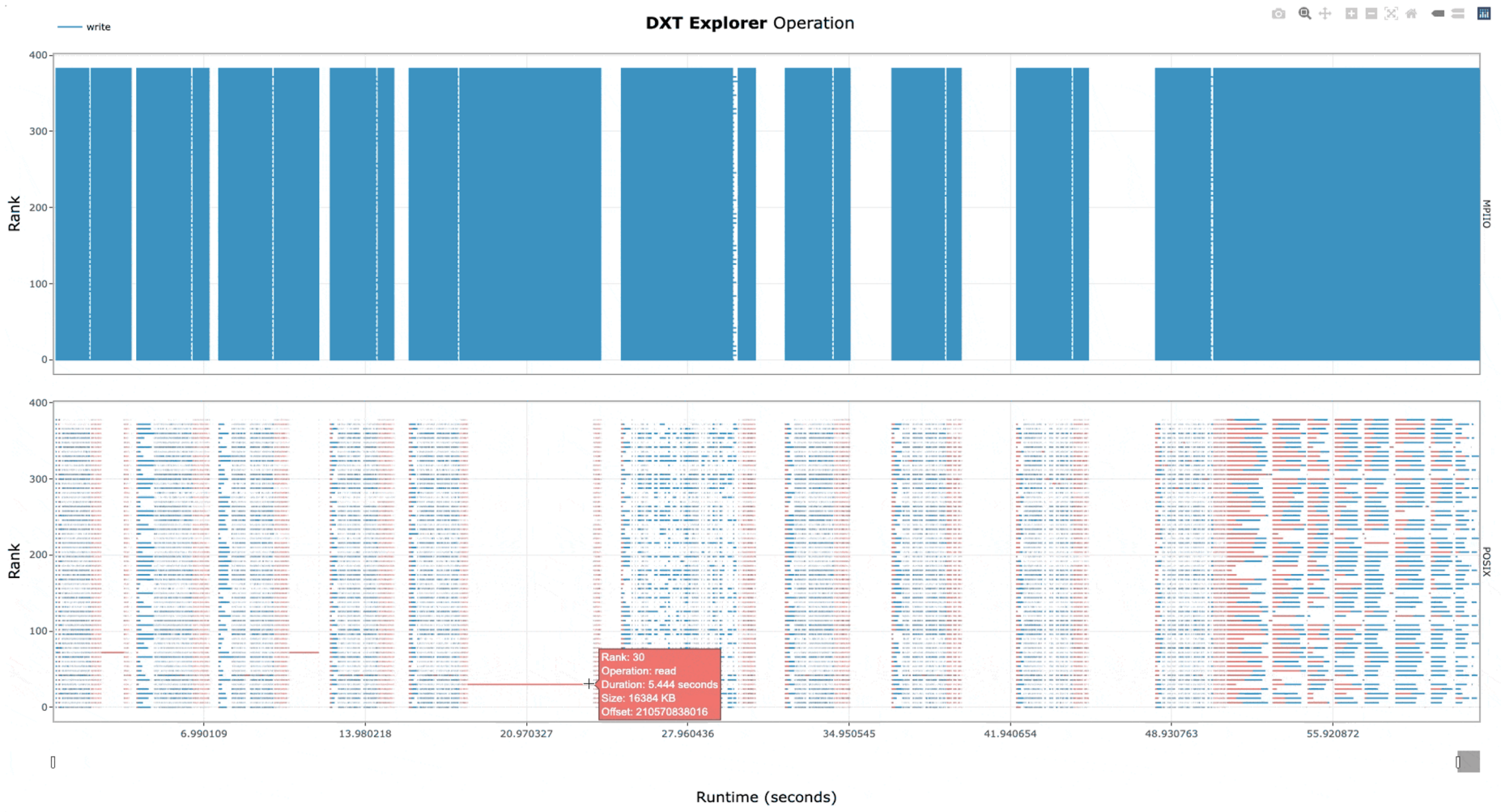
```
usage: dxt-explorer [-h] [-o OUTPUT] [-t] [-s] [-d] [-l] [--start START] [--end END] [--from START_RANK] [--to END_RANK] darshan

DXT Explorer:

positional arguments:
  darshan               Input .darshan file

optional arguments:
  -h, --help            show this help message and exit
  -o OUTPUT, --output OUTPUT
                        Name of the output file
  -t, --transfer        Generate an interactive data transfer explorer
  -s, --spatiality      Generate an interactive spatiality explorer
  -d, --debug           Enable debug mode
  -l, --list            List all the files with trace
  --start START         Report starts from X seconds (e.g., 3.7) from beginning of the job
  --end END             Report ends at X seconds (e.g., 3.9) from beginning of the job
  --from START_RANK     Report start from rank N
  --to END_RANK         Report up to rank M
```

Explore I/O operations by zooming in



Explore the timeline by **zooming in and out** and observing how the **MPI-IO** calls are translated to the **POSIX** layer. For instance, you can use this feature to detect stragglers.

Hover over to find contextual informaion

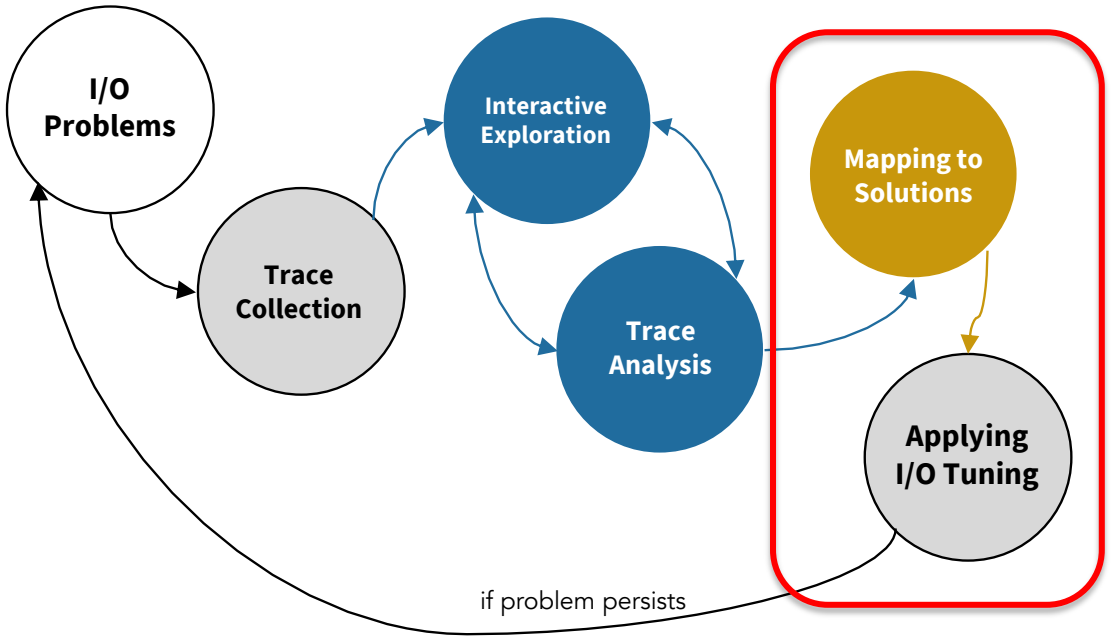


Visualize relevant information in the **context** of **each I/O call** (rank, operation, duration, request size, and OSTs if Lustre) by hovering over a given operation.

Jean Luca Bez, Houjun Tang, Bing Xie, David Williams-Young, Rob Latham, Rob Ross, Sarp Oral, and Suren Byna, "I/O Bottleneck Detection and Tuning: Connecting the Dots using Interactive Log Analysis", 6th International Parallel Data Systems Workshop (PDSW) 2021, held in conjunction with SC21

Process to find and solve I/O problems

- Collect logs / traces
- Analyze for finding problems
- Find tuning options
- Apply tuning solutions



Common causes of poor I/O performance

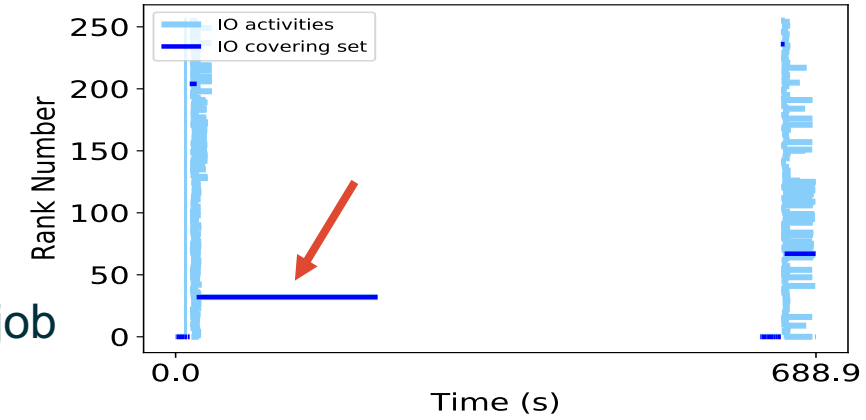
- One rank undertaking excessive workload is a common I/O bottleneck
- Using default stripe count of 1 causing I/O bottleneck (on Lustre)
- Unbalanced IO distribution across storage servers
- Transient system weather change.
- Frequent I/O phases
- Read-after-write yielding good I/O bandwidth

Application-level analysis - Quantum1 app

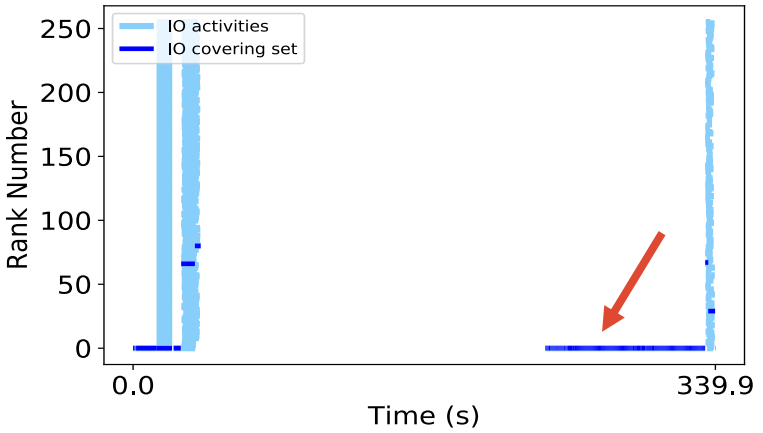
- One user's job's I/O bandwidth spans across 4 ranges, and the I/O bottlenecks of each range is not directly perceivable from the parallel coordinate plot.

Job-Level Analysis of Quantum1 IO

(0, 1]GB/s job



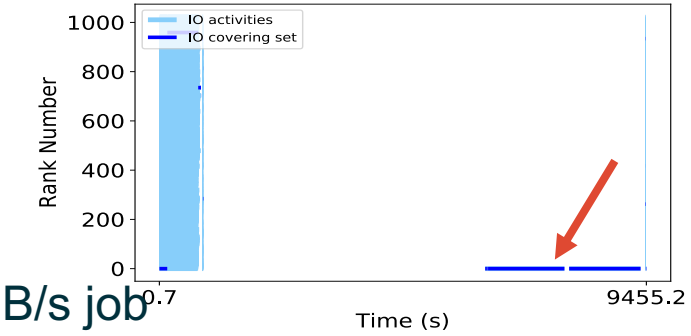
One rank IO suffers from transient metadata load change



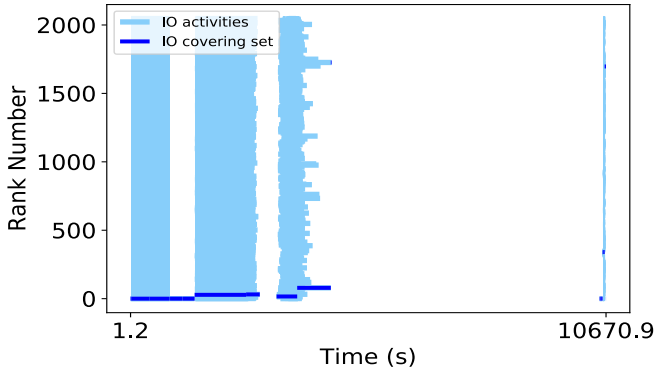
(1, 10]GB/s job

Rank 0 writes a large number of configuration files

(10, 100]GB/s job



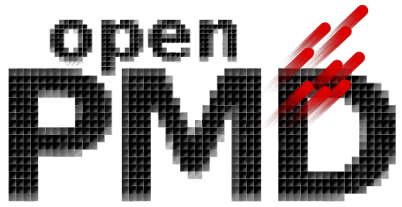
Rank 0 takes more time than others due to its larger workload, but bandwidth is still good as other ranks also write a lot of data



(100, 1000]GB/s job

There are four I/O phases, each I/O phase follows read after write pattern

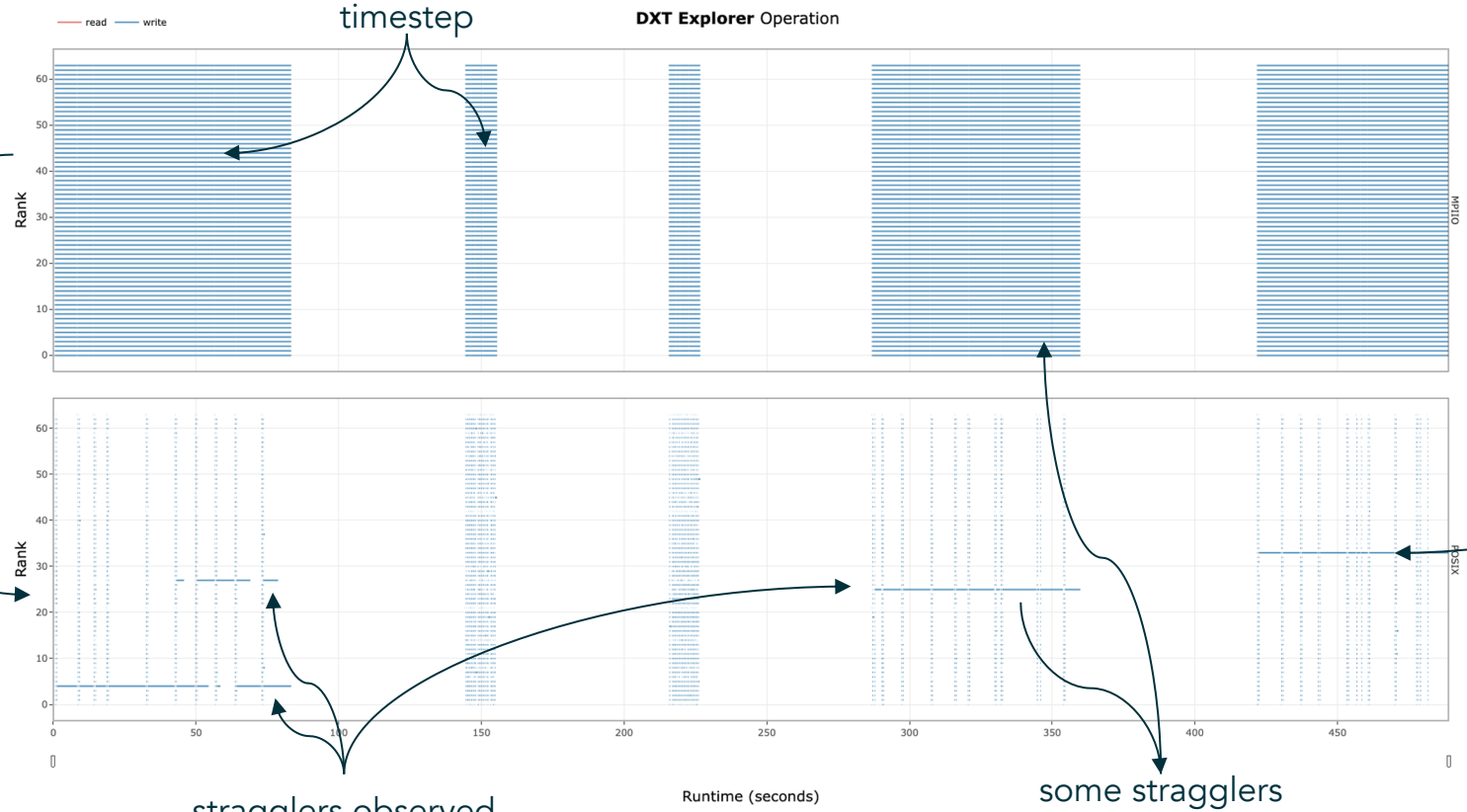
What can we see with OpenPMD?



github.com/hpc-io/dxt-explorer

same amount of data in each timestep

collective calls translate into several POSIX calls



Rank: 25
Operation: write
Duration: 12.07 seconds
Size: 32768 KB
Offset: 16273899520

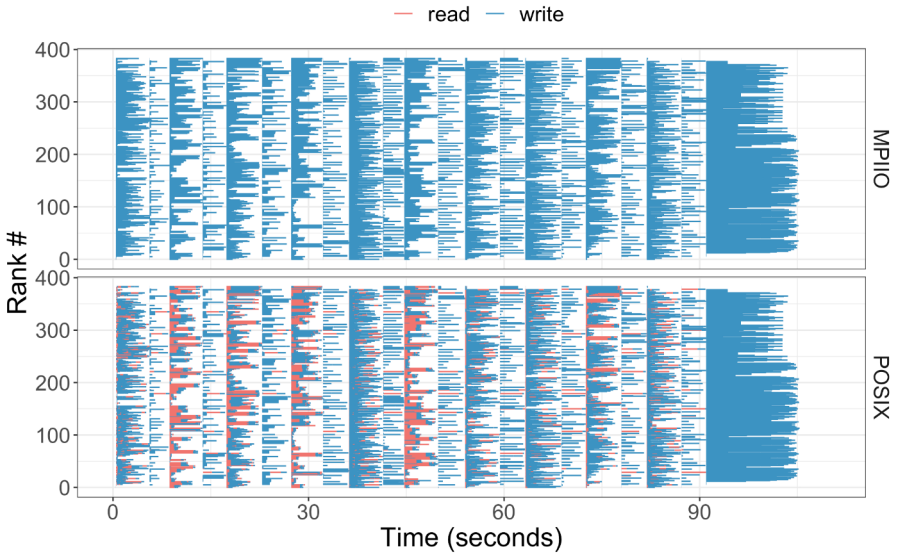
stragglers observed in different ranks

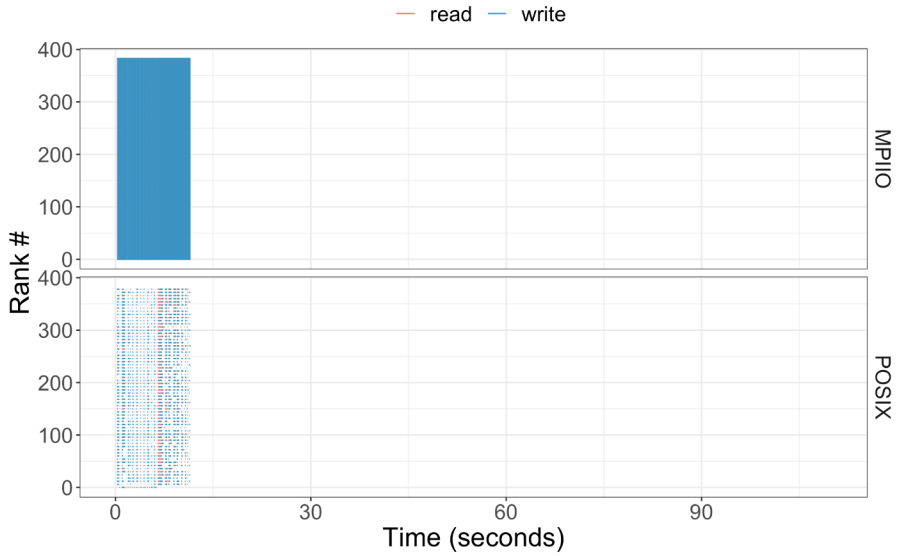
some stragglers make the collective calls take longer

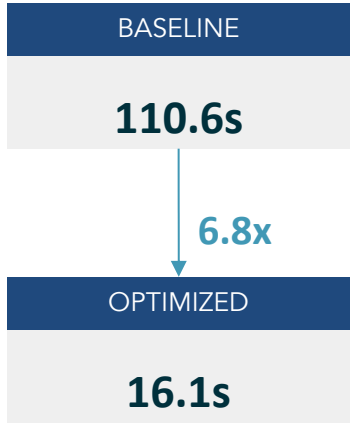
OST information will show up if available (e.g. Lustre)

OpenPMD use case

- Collective I/O using ROMIO: **1.54x** speedup
- GPFS large block I/O + HDF5 collective metadata: **+3.8x** speedup
  - Discovered an **issue** with collective metadata introduced in HDF5 1.10.5
- Fix combined with previous optimizations gives a total of **6.8x** speedup from baseline

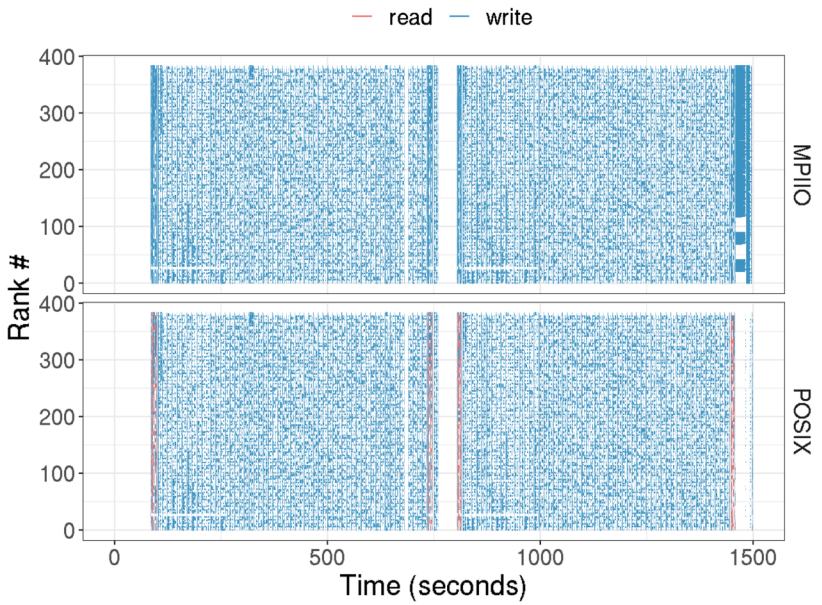


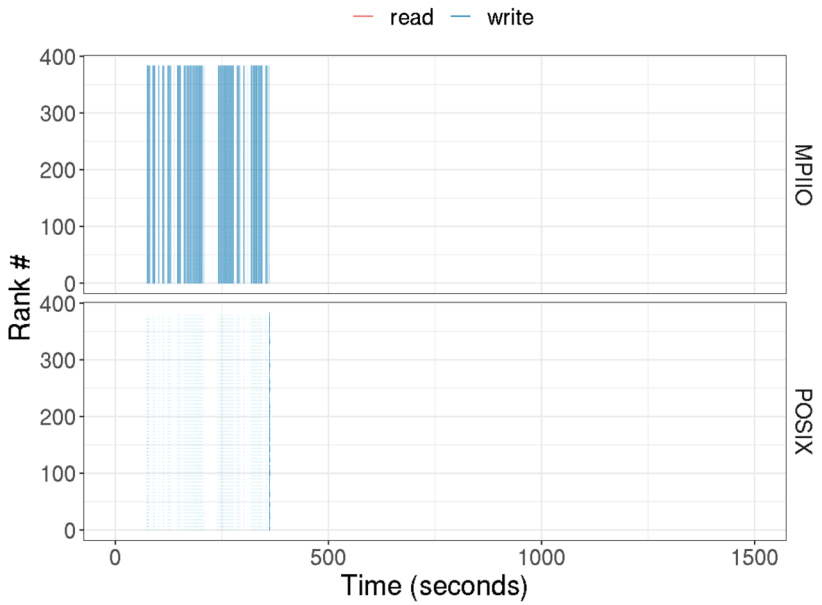


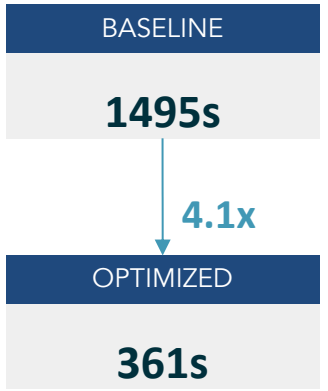


FLASH use case

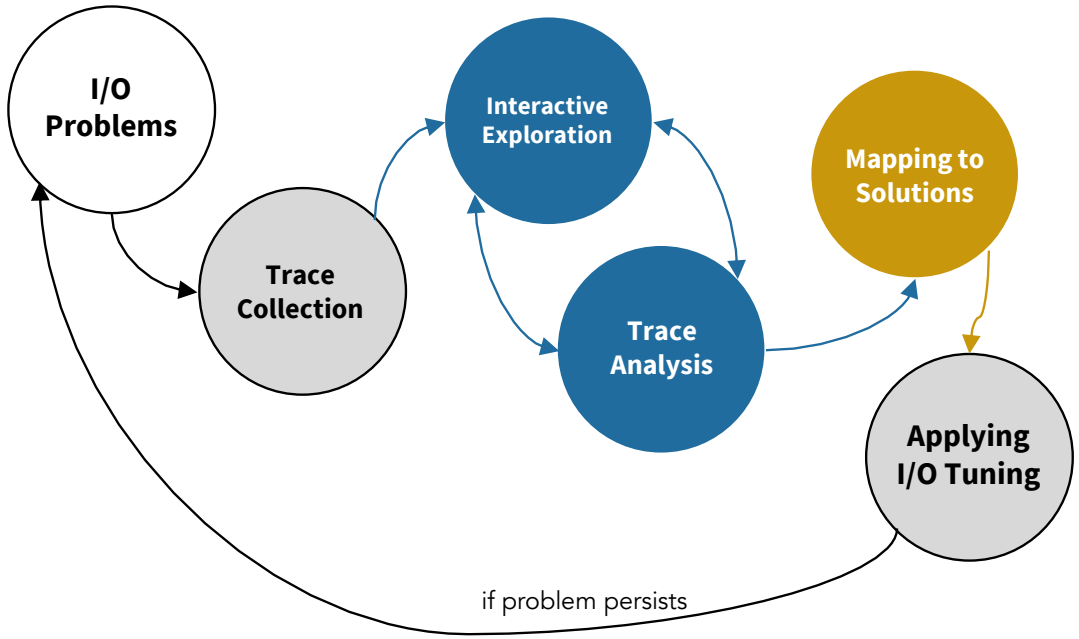
- 2 checkpoint files (≈2.3TB each) and 2 plot file (≈14GB each)
- FLASH was not using collective MPI-IO calls
- **Optimizations**: collective I/O, HDF5 alignment, and defer metadata flush





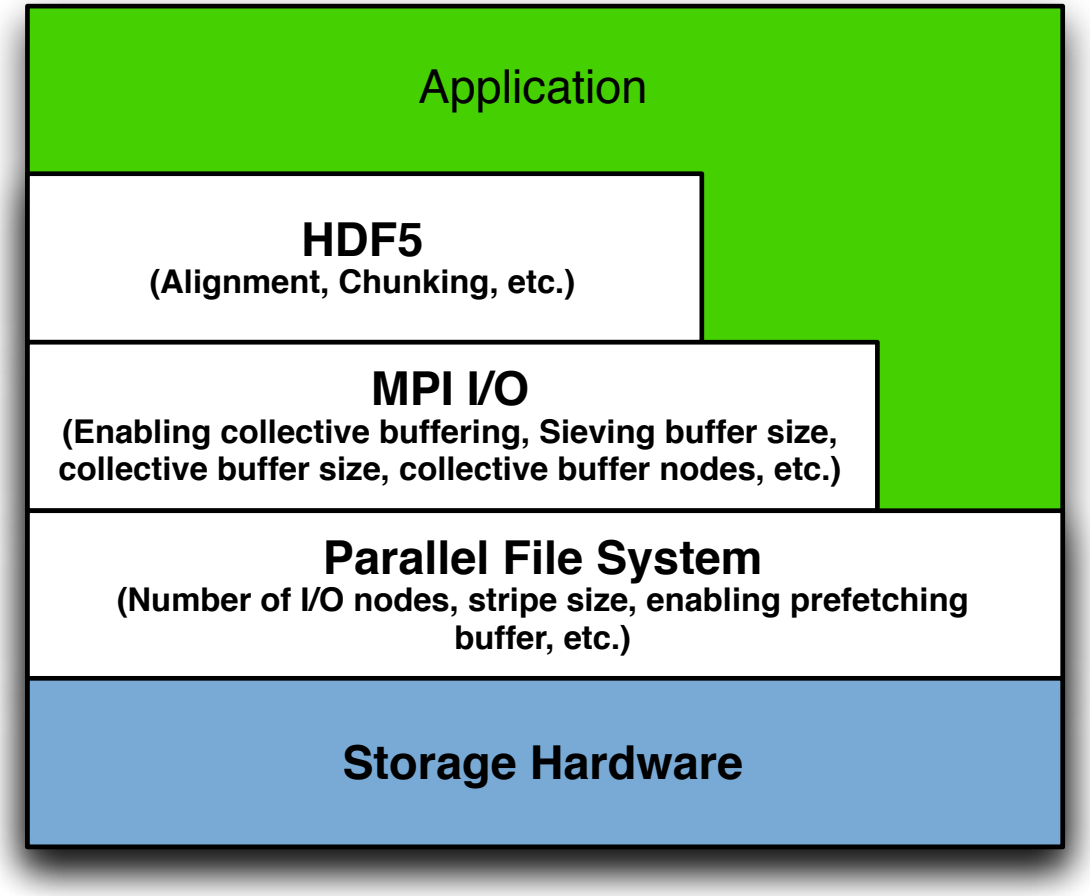


How to automate the tuning process?



Challenges in auto-tuning I/O performance

- Options for performance optimization
- Complex inter-dependencies among layers
- Performance is dependent on application, HPC platform, and problem size/concurrency
- Achieving peak I/O performance often requires significant I/O subsystem expertise

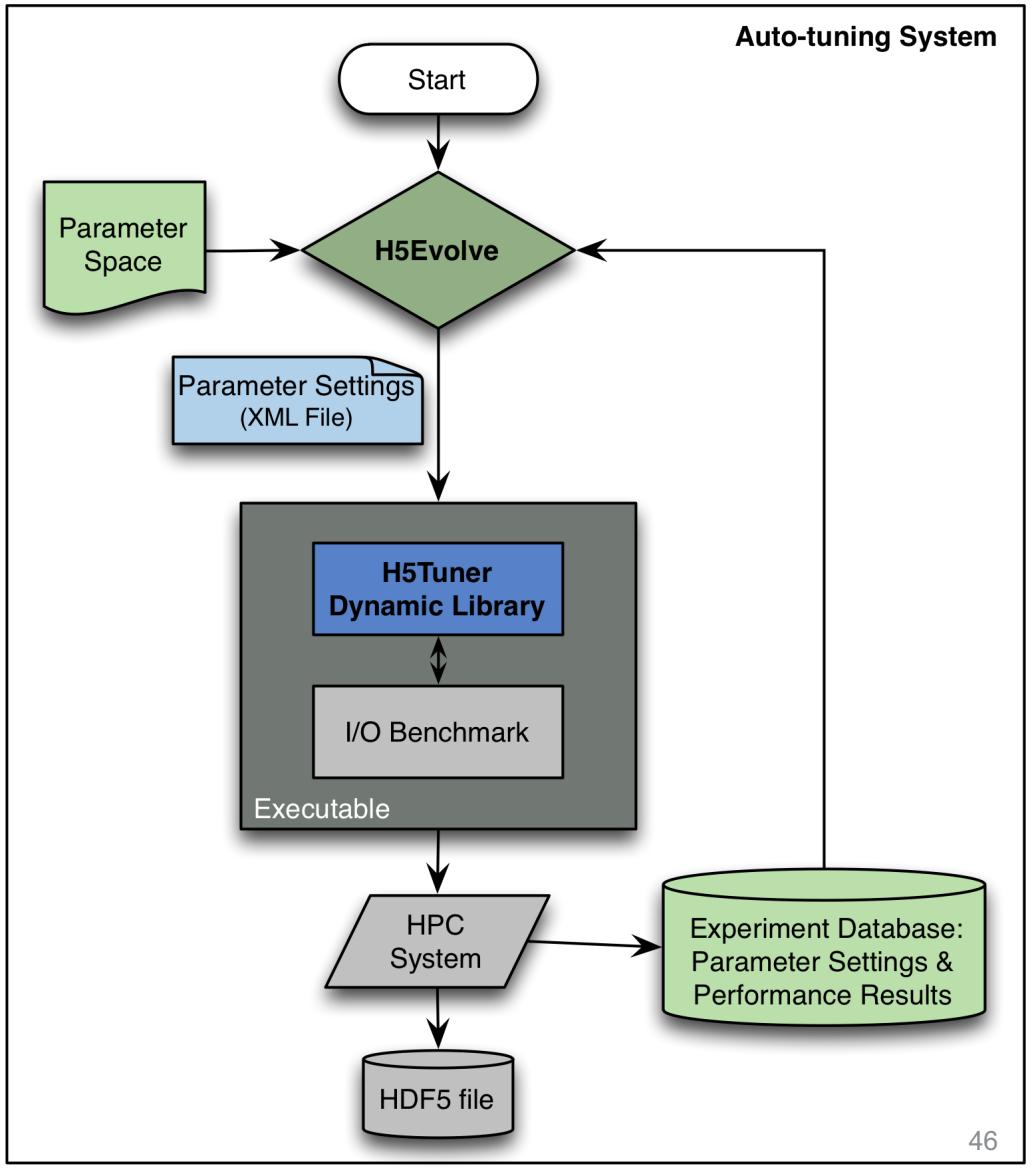


Tuning combinations are abundant

- A sample space of optimization parameters
  - HDF5 alignment: (1,1) to (16KB, 32KB) → 14 samples
  - Lustre stripe count: 1 to 156 → 10 samples
  - Lustre stripe size = MPI collective buffer size: 1MB to 128MB → 8 samples
  - MPI Collective buffer nodes: 1 to 256 → 12 samples
  - Total search space: > 13,000 combinations
- Searching through all combinations manually is impractical
- Users, typically domain scientists, should not be burdened with tuning

I/O auto-tuning

- Auto-tuning framework to search the parameter space with reduced number of combinations
- HDF5 I/O library sets the optimization parameters
- H5Tuner: Dynamic interception of HDF5 calls
- H5Evolve: Genetic algorithm based selection



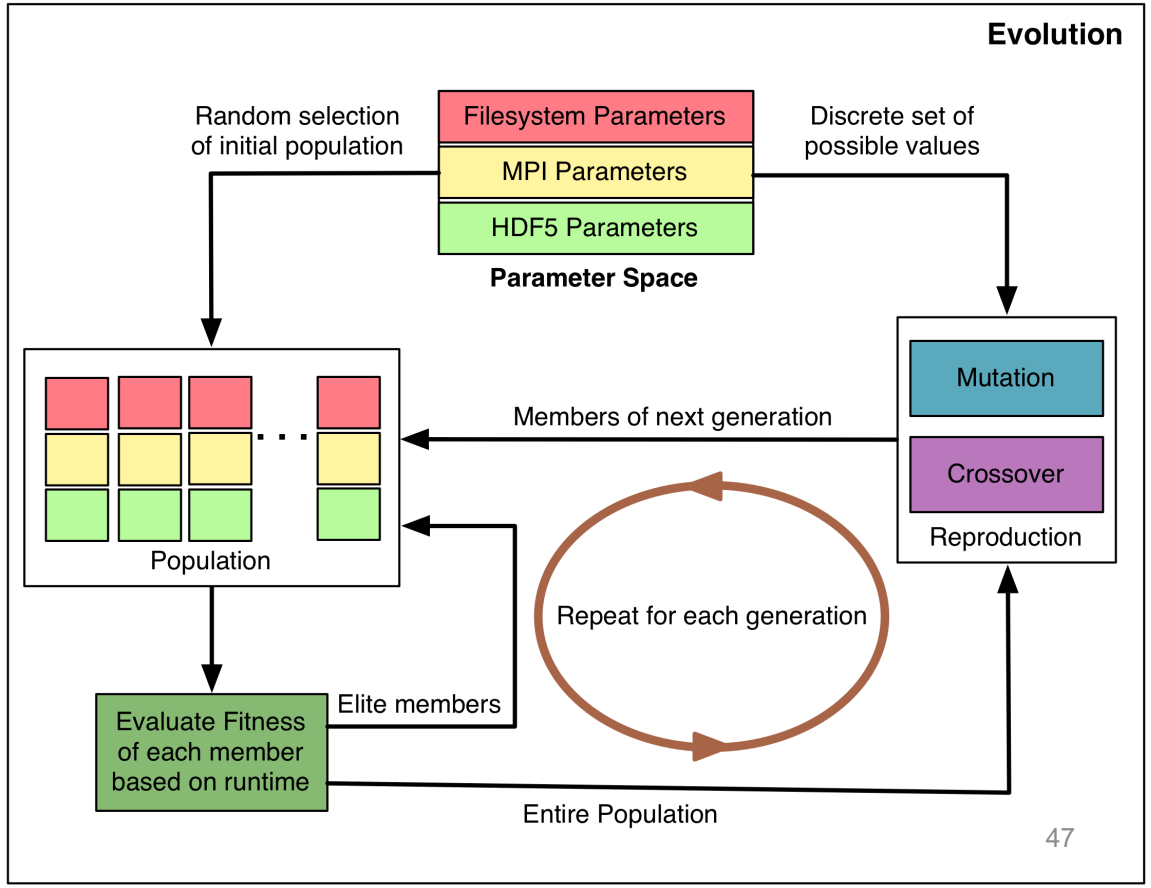
B. Behzad, H. Luu, J. Huchette, S. Byna, Prabhat, R. Aydt, Q. Koziol, and M. Snir, "Taming Parallel I/O Complexity with Auto-Tuning", ACM/IEEE Supercomputing 2013 (SC13)

Genetic Algorithm-based search space

- GA evaluates fitness (I/O performance) and selects members based on least runtime and on mutation of various optimization parameters

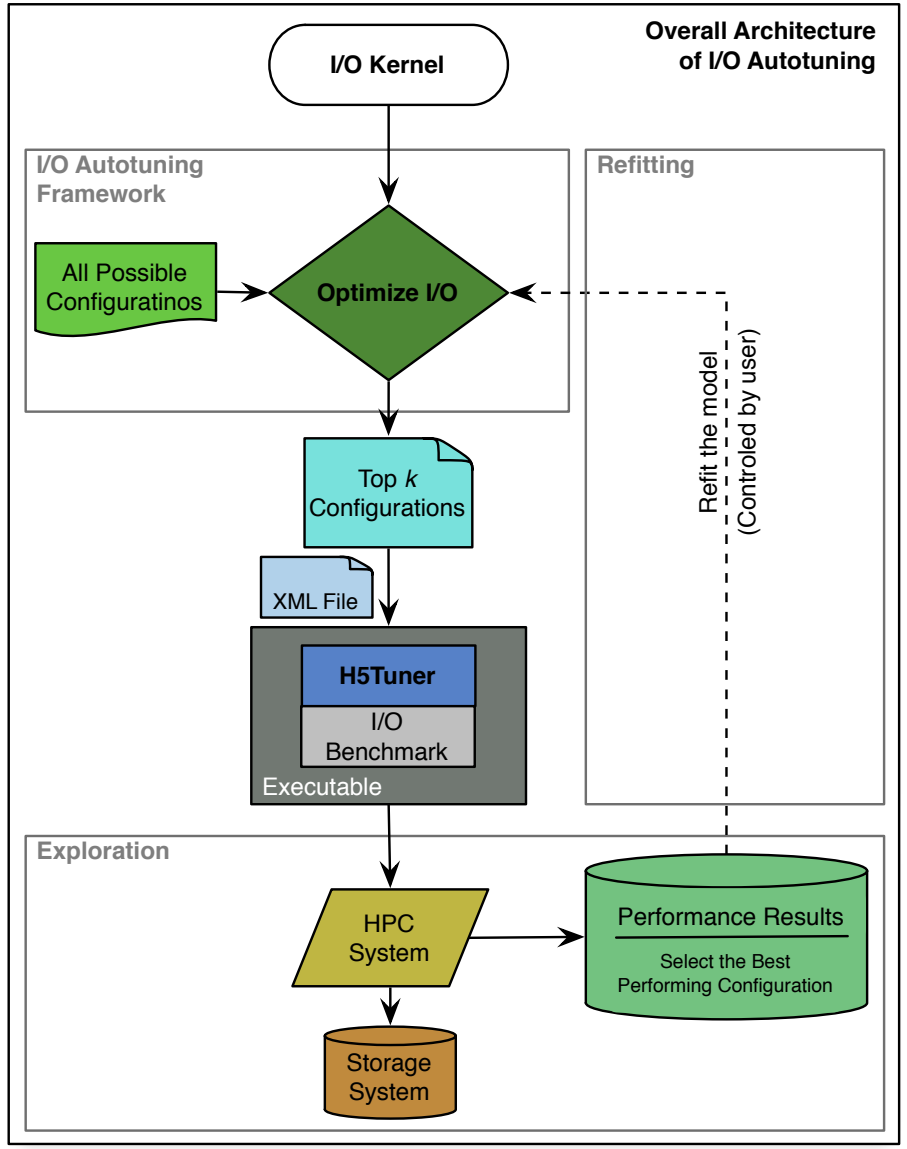
- Problems
  - Long search time

    (more than 12 hours)
- Limited general purpose applicability for different problem sizes



Model-driven Auto-tuning

- Replaced GA-based H5Evolve with I/O performance predictor
  - Using empirical models of the I/O performance
- Search Strategy
  - Step 1 (Pruning): Use an empirical model to reduce the search space of interest
  - Step 2 (Exploration): Search a smaller space
  - Step 3 (optional): Refit the model and repeat Steps 1 and 2



Babak Behzad, Suren Byna, Stefan Wild, Prabhat and Marc Snir, "Dynamic Model-driven Parallel I/O Performance Tuning", IEEE Cluster 2015

Performance Model

- Non-linear regression model

$$\mathrm{m}(x;\beta) = \sum_{k=1}^{n_b} \beta_k \phi_k(x)$$

- Linear combinations of n_b non-linear basis functions (ϕ_k), and hyper-parameters β (selected by standard regression approach) for a parameter configuration of x

$$\mathrm{m}(x) = \beta_1 + \beta_2 f + \beta_3 \frac{f}{a} + \beta_4 \frac{a}{c} + \beta_5 \frac{a}{s} + \beta_6 \frac{cs}{a} + \beta_7 \frac{cf}{a}$$

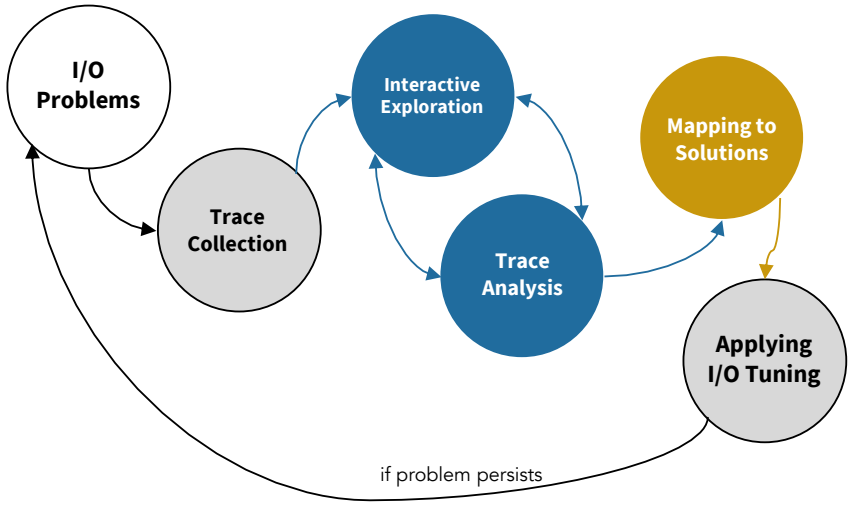
- f: file size
- a: number of aggregators
- c: stripe count
- s: stripe size

Performance Results – VPIC-IO

# of cores	File Size (GB)	Modeling Bandwidth (MB/s)	GA Bandwidth (MB/s)	Default Bandwidth (MB/s)	Speedup
128	32	2075	3034	472	**4.4X**
512	128	5185	-	409	**12X**
1024	256	6182	-	337	**18X**
2048	512	11422	14900	412	**28X**
4096	1024	14892	17620	365	**41X**
8192	2048	18857	-	345	**54X**

Future: Automating the tuning process

- Trace collection
  - Dynamic adjustment of tracing granularity at runtime
- Trace analysis and problem identification are currently manual
  - Need pattern detection for performance issuess
- Application of I/O tuning options is still manual
  - I/O libraries should allow application of tuning parameters at runtime
  - File systems could be designed to be adjusting to traffic from many applications



Conclusions

- I/O problems can be due to file system issues or application issues
  - Transient or long-term issues

- Automatic tuning is complex due to interdependencies among multiple software layers
  - Too many combinations

- Open research problem - automating I/O performance tuning

Contact: Suren Byna (SByna@lbl.gov)