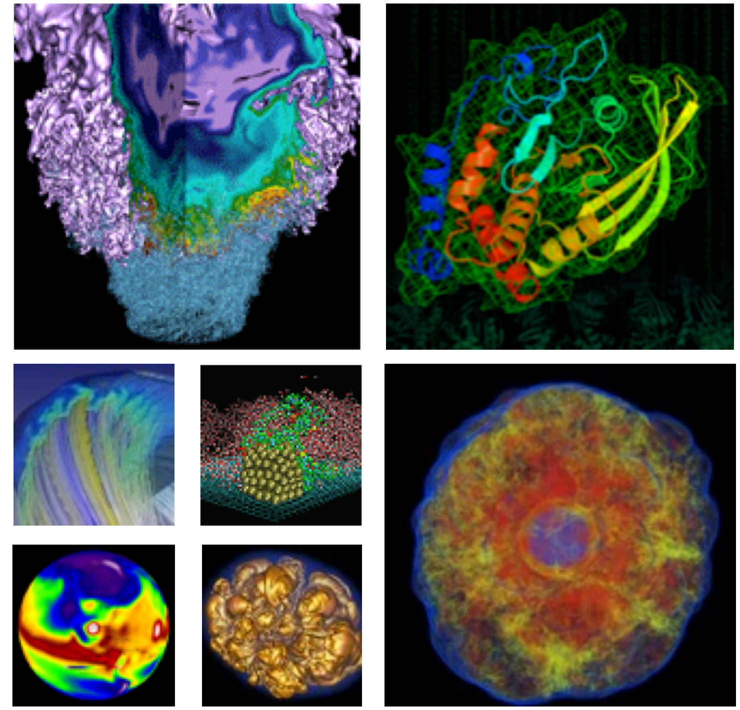
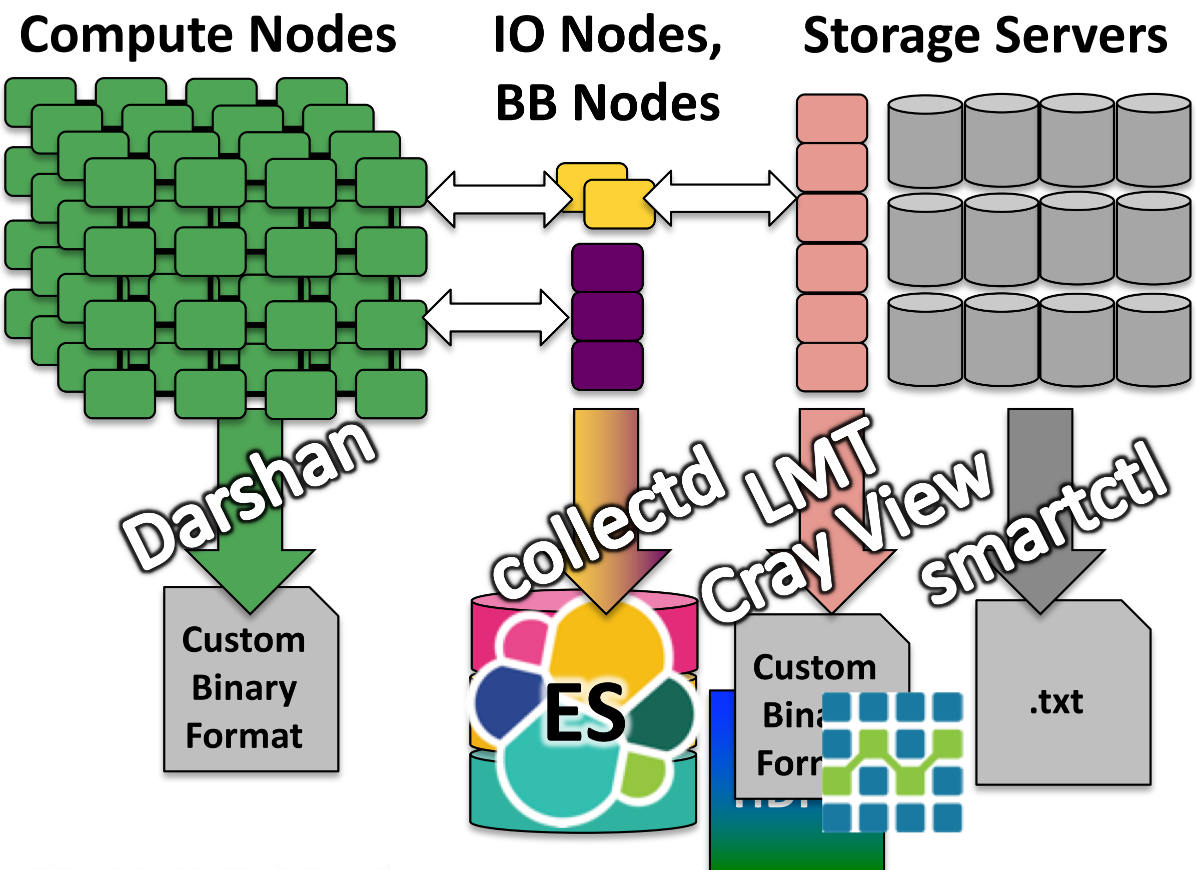
TOKIO on ClusterStor: Connecting Standard Tools to Enable Holistic I/O Performance Analysis



Glenn K. Lockwood, Shane Snyder, George Brown, Kevin Harms, Philip Carns, Nicholas J. Wright

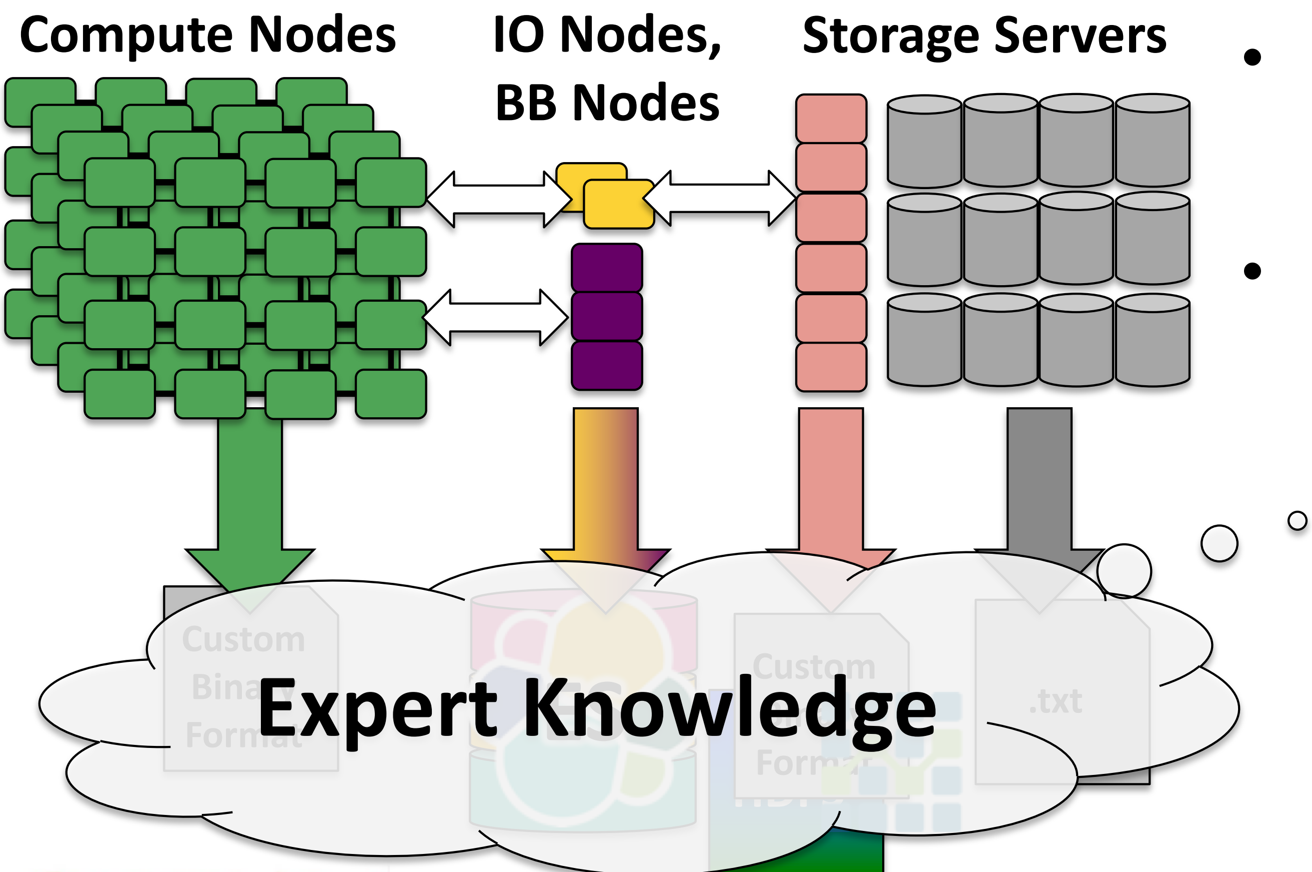
May 22, 2018

Understanding I/O today is hard



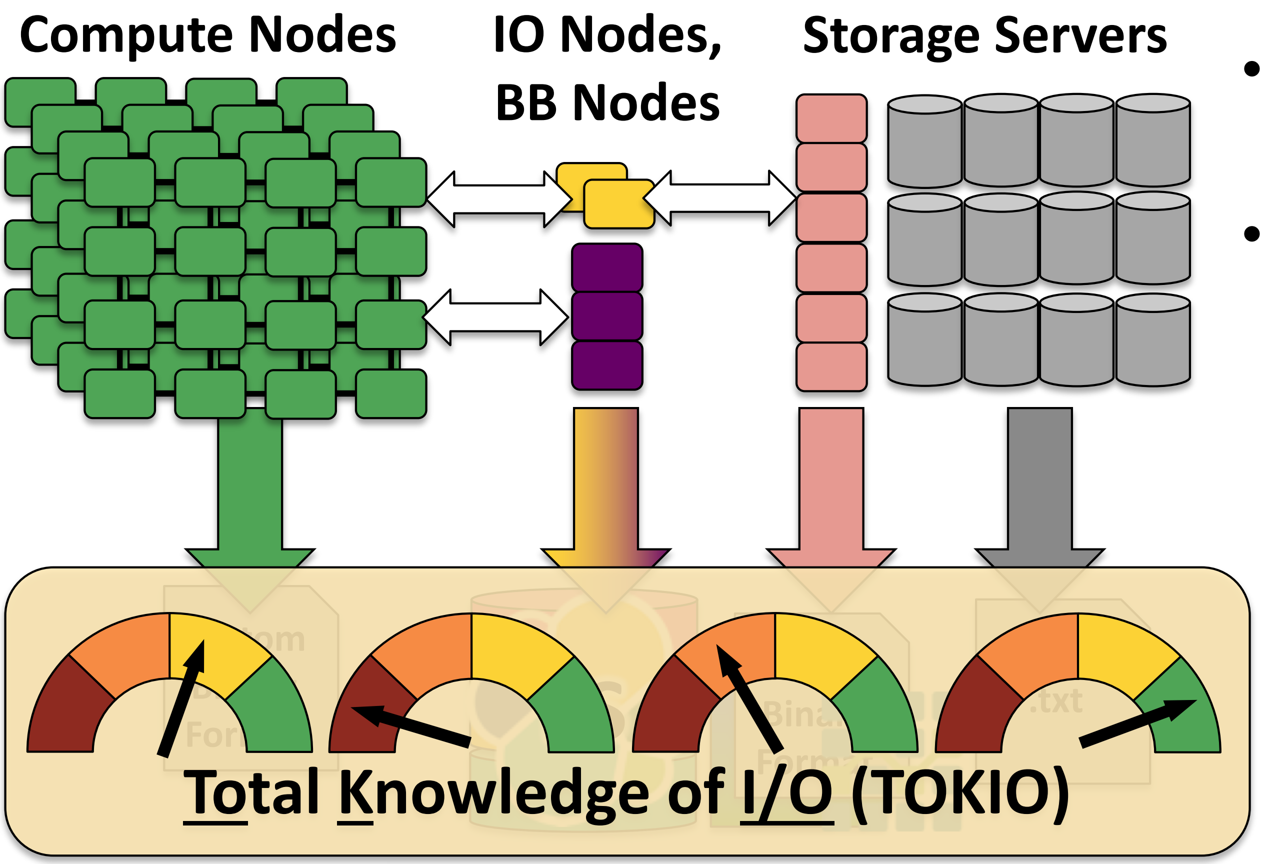
- Storage hierarchy is getting more complicated
- Currently monitoring each component separately is standard practice

Understanding I/O today is hard



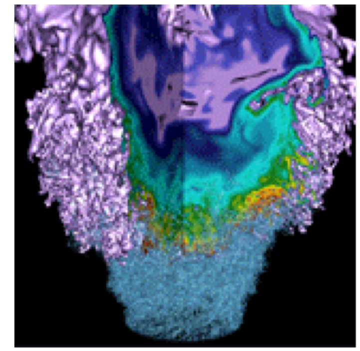
- Storage hierarchy is getting more complicated
- Currently monitoring each component separately is standard practice

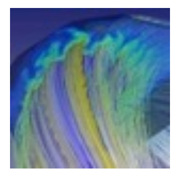
Total Knowledge of I/O with holistic analysis

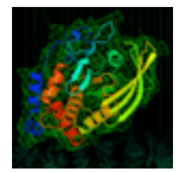


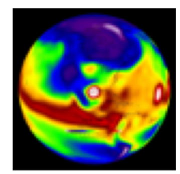
- TOKIO augments expert knowledge
- TOKIO enables greater insight from *existing* data sources
  - Combines and normalizes component-level data
  - Provides integrated analysis and a holistic view

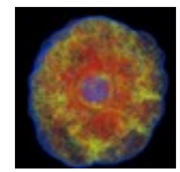
Designing a Framework for Total Knowledge of I/O

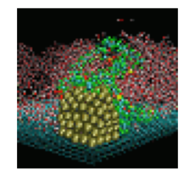








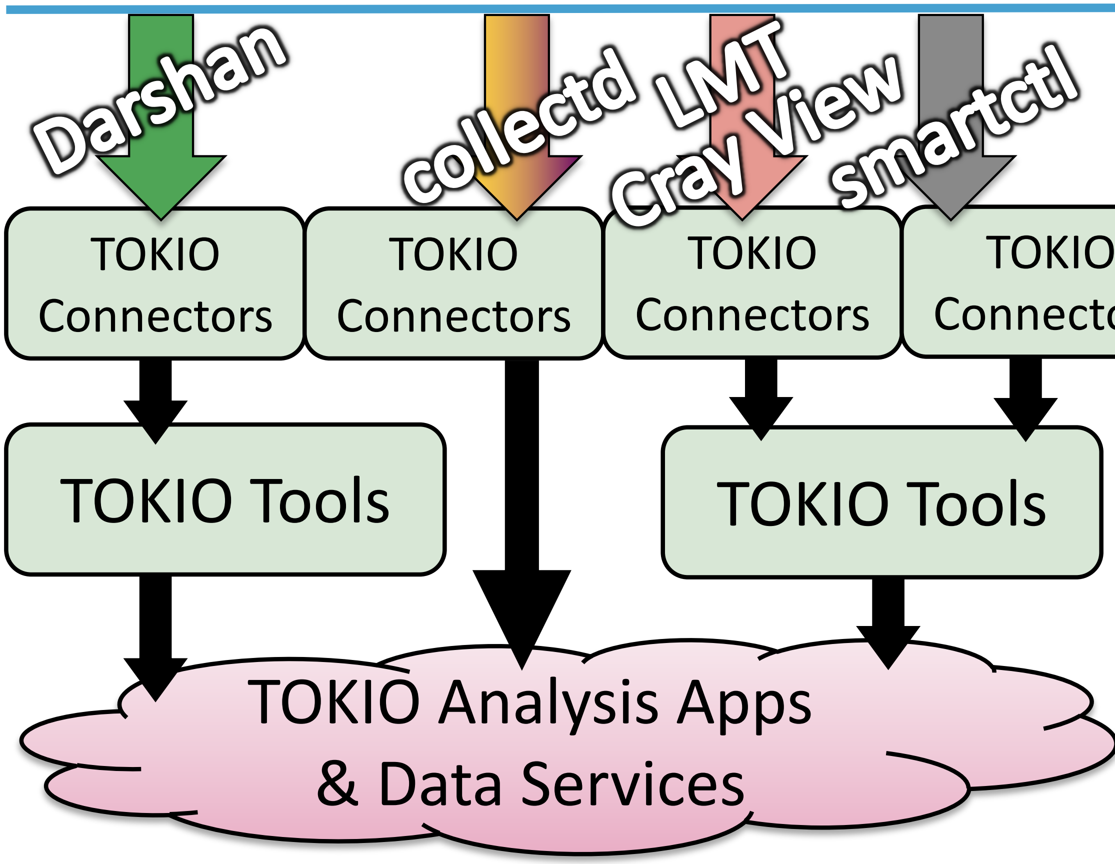




Designing an I/O monitoring framework

1. **Use existing tools already in production** → Interact with tools, don't reinvent them!
2. **Leave data where it is** → Don't force all data into a single datastore
3. **Make data as accessible as possible** → Instead, use software interfaces and abstractions to normalize data

Three layers of TOKIO

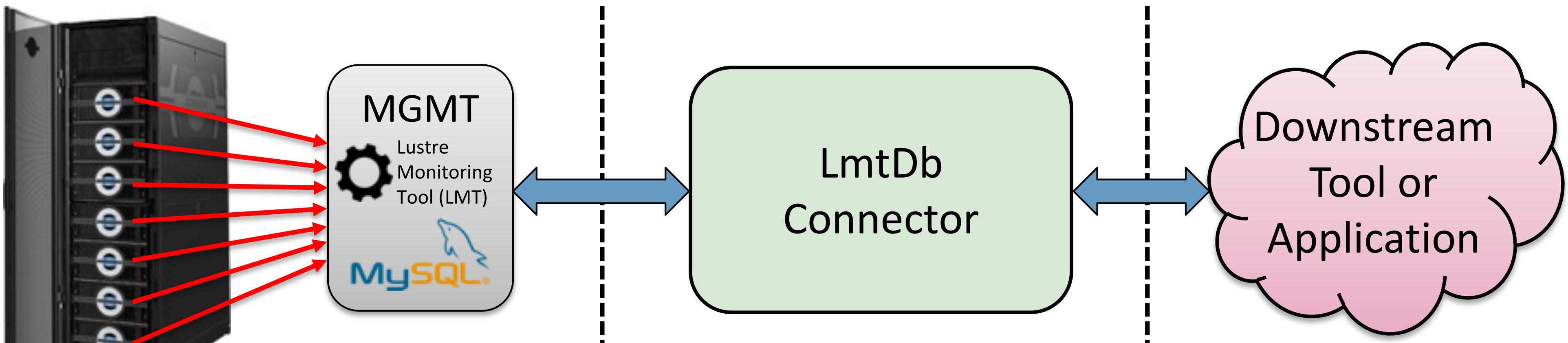


Connectors translate tool-specific output into language-specific objects

Tools provide more convenient, portable programming interfaces

Analysis apps turn data into insight

pytokio connectors talk to monitoring tools



LMT included in Cray ClusterStor software stack

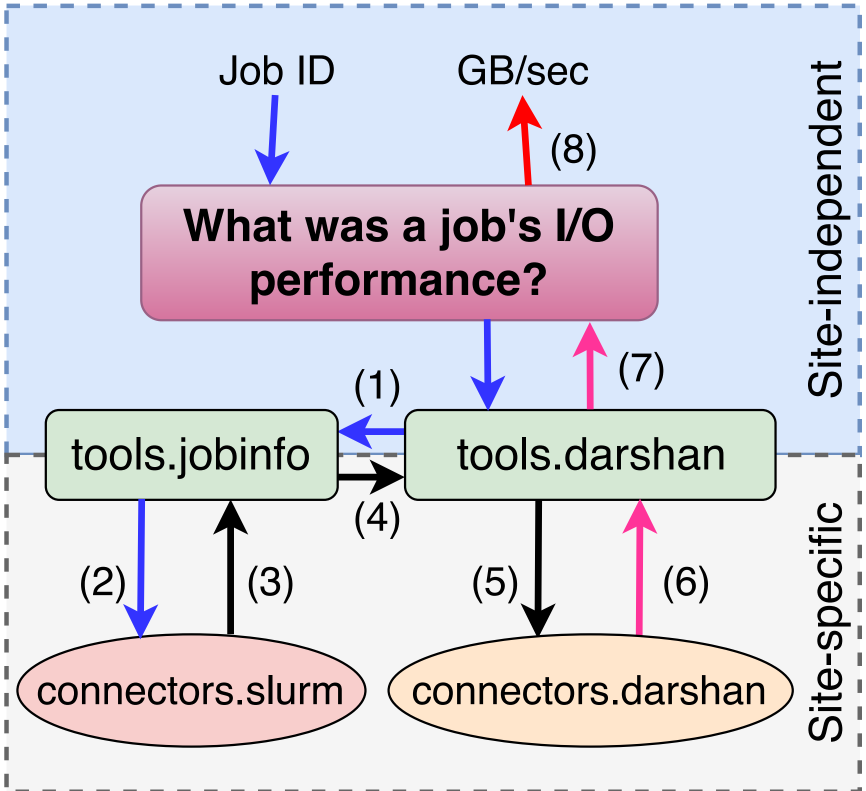
pytokio connector translates high-level functions (“get_ost_data”) into MySQL queries, and returns Python objects

Downstream app receives tuples, dicts from pytokio connector

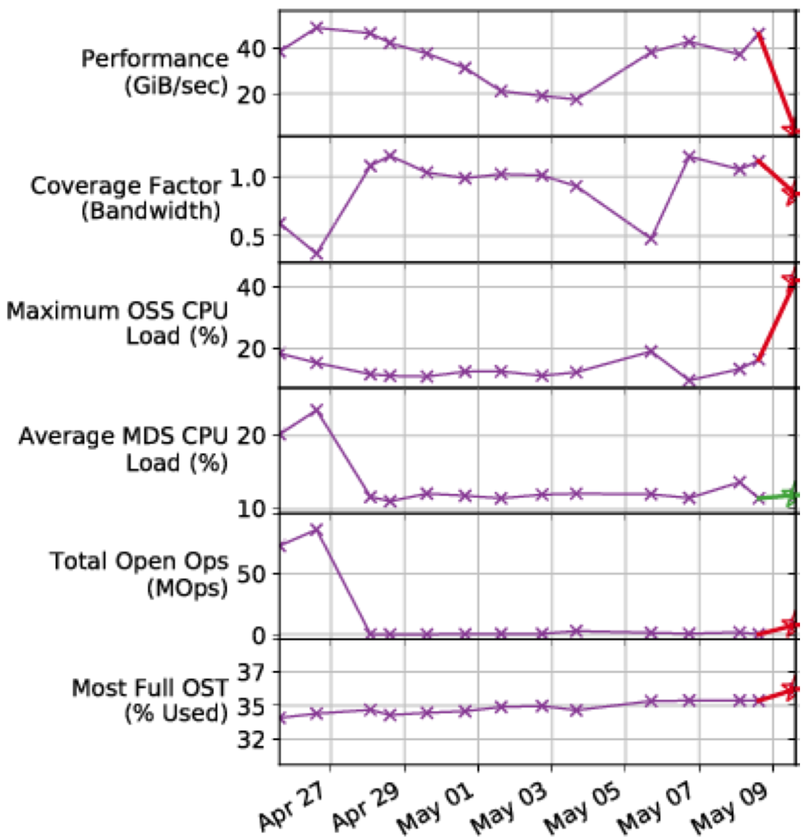
pytokio tools provide convenience and portability

Expose higher-level functions to make programming easier/portable

- Darshan tool needs job start date to find darshan logs
- Jobinfo tool turns job id into start date using site-specific connector (Slurm, PBSpro)
- Darshan tool opens all relevant Darshan logs and returns performance data



Analysis apps provide insight



Unified Monitoring and Metrics Interface shows holistic performance in a single pane

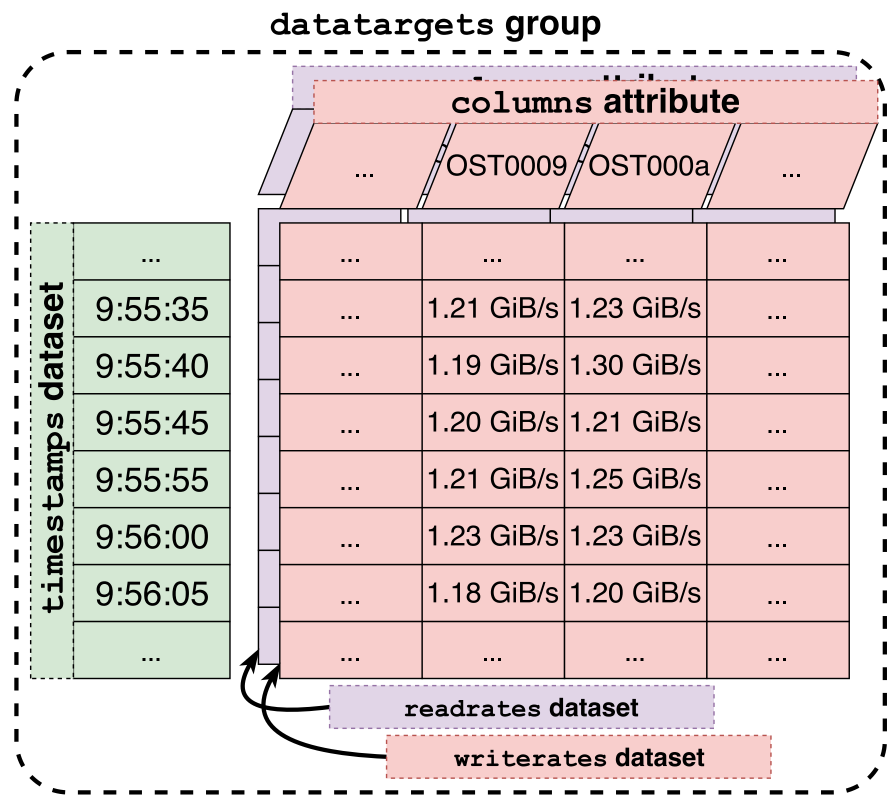
- BDCATS run on ALCF's Cray XC-40, Theta over 14 days
- Performance on May 11 was bad due to data and metadata contention

G. K. Lockwood et al, "UMAMI: A Recipe for Generating Meaningful Metrics through Holistic I/O Performance Analysis," in Proceedings of the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Computing Systems - PDSW-DISCS '17. 2017, pp. 55–60.

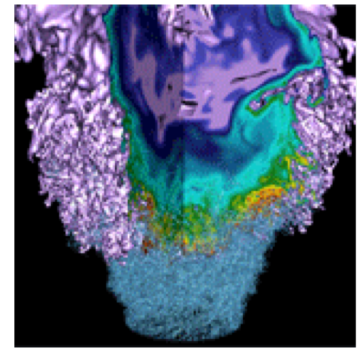
Data services help manage data

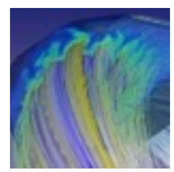
Store real-time operational data for retrospective analysis

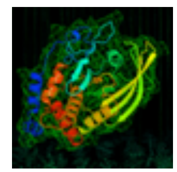
- **TOKIO Time Series files**
  - File system-agnostic format and API
  - Indexed by time and storage device
  - Based on HDF5
- **pytokio Archival Services** use this format
  - Lustre load data (LMT connector)
  - DataWarp load data (collectd + Elasticsearch connector)

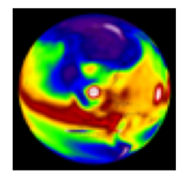


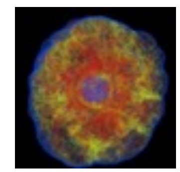
Solving I/O problems with example pytokio analysis apps

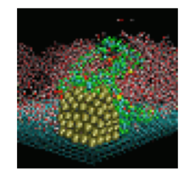




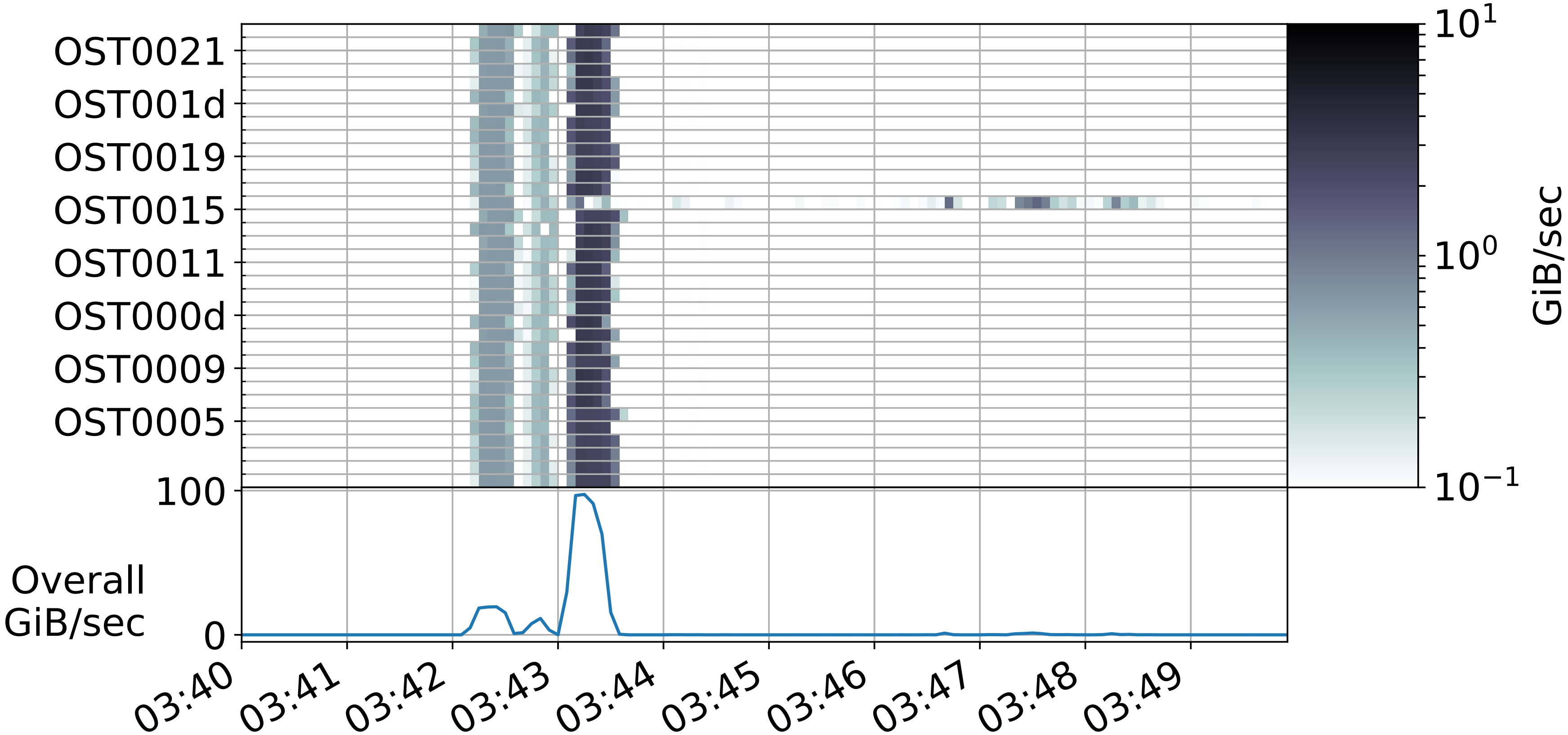




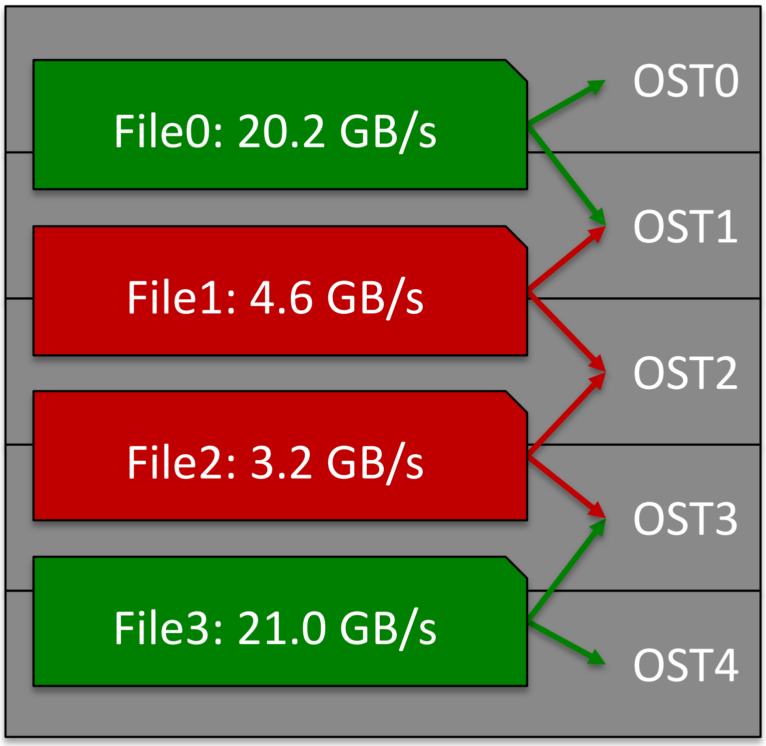




Straggling OSTs are a common problem

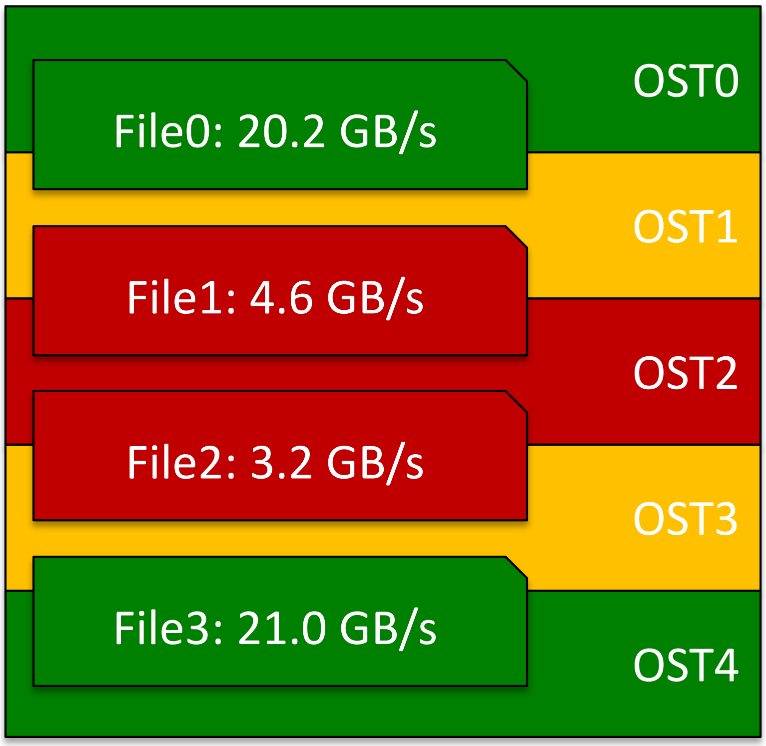


darshan_bad_ost finds straggling OSTs



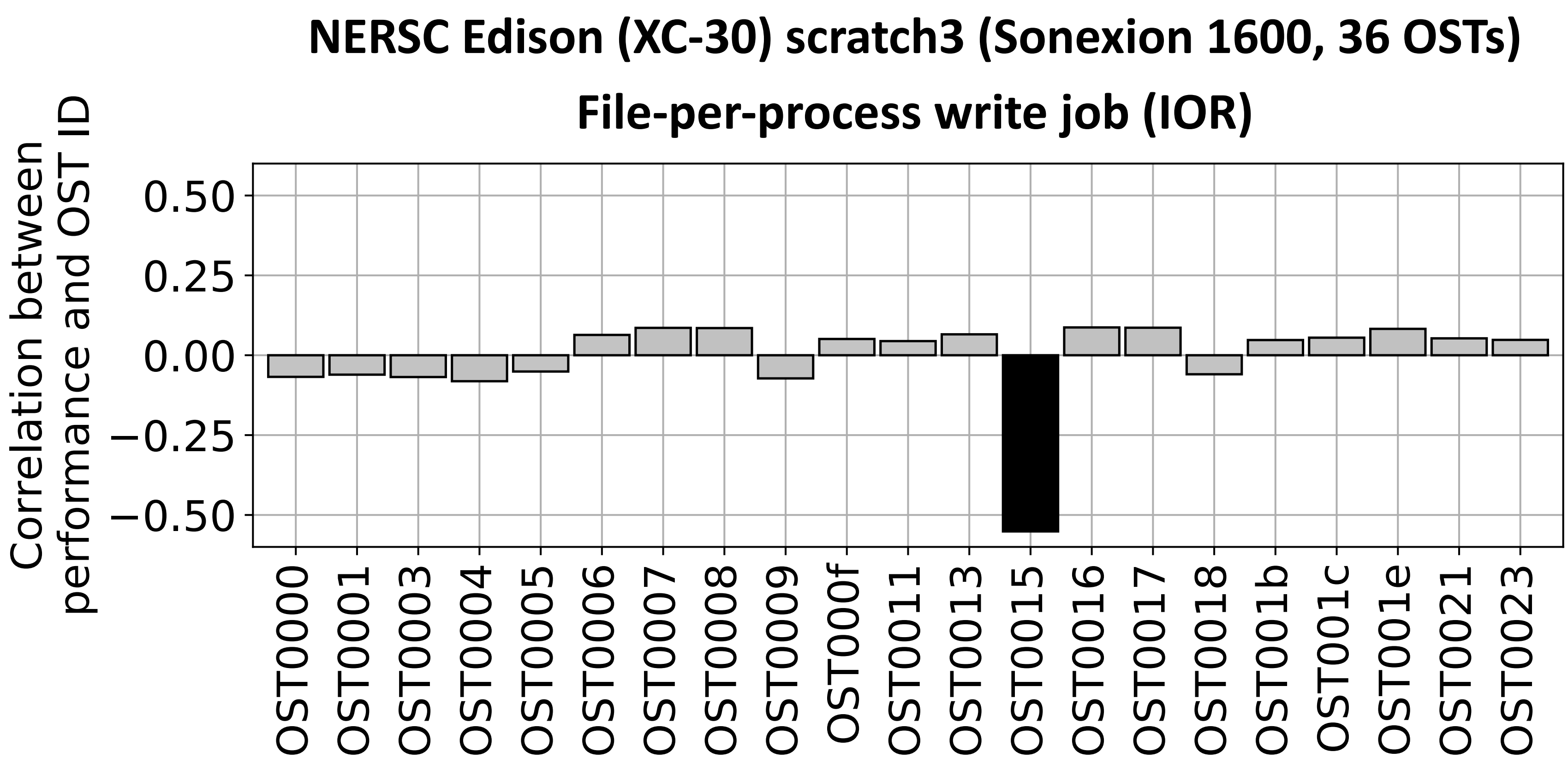
- **Application I/O is slow**
- Darshan shows some files wrote out very slowly
- Which OSTs did those files touch?
- Do any OSTs consistently correlate with slow I/O?

darshan_bad_ost finds straggling OSTs

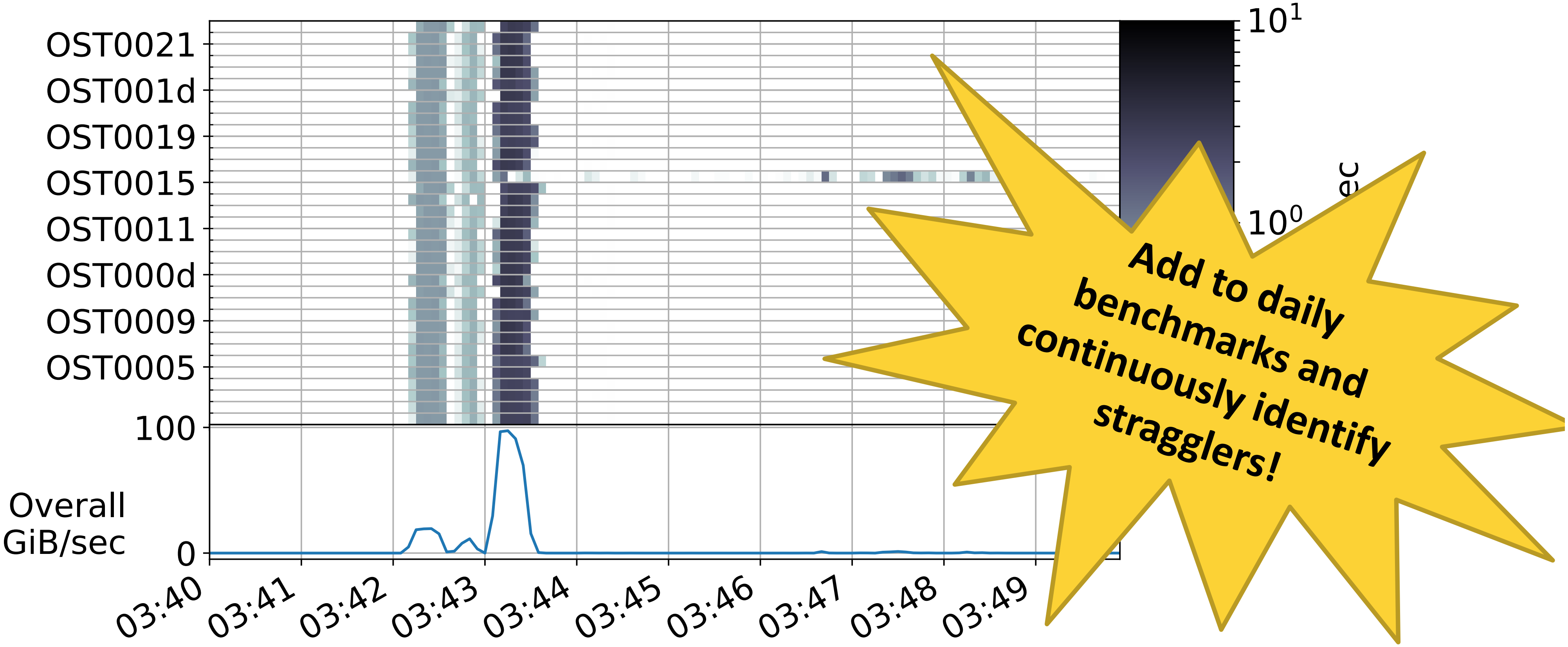


- **Application I/O is slow**
- Darshan shows some files wrote out very slowly
- Which OSTs did those files touch?
- Do any OSTs consistently correlate with slow I/O?

darshan_bad_ost on Edison

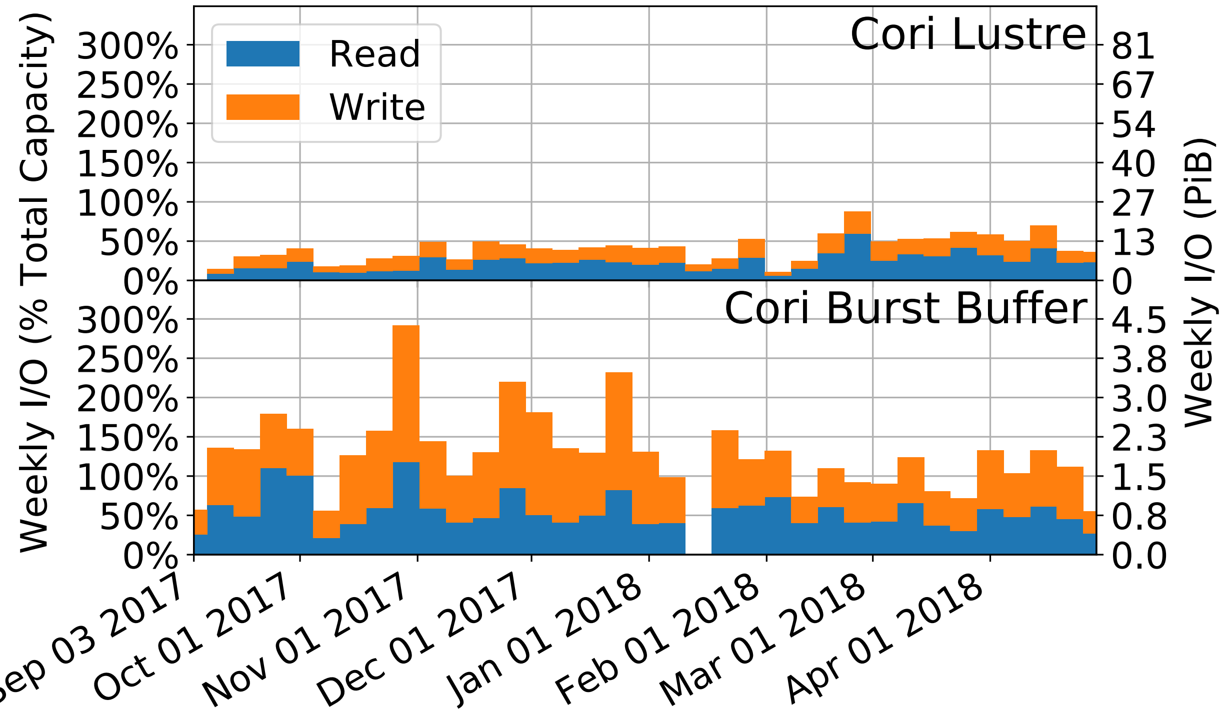


darshan_bad_ost on Edison



Optimizing I/O workload with DataWarp

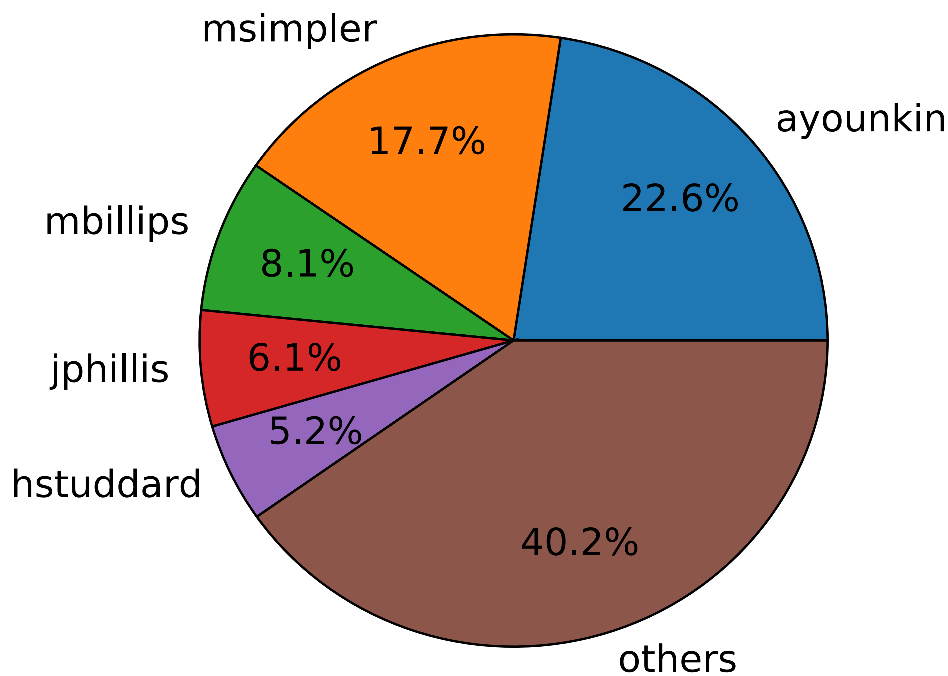
- Burst buffer should be busy, but not too busy
- Burst buffer is not close to its endurance limit (~100x below limit)
- Cori Lustre sees a lot of I/O traffic, most is temporary data
- What workloads can we move from Lustre to DataWarp?



Data from LMT connector (TOKIO Time Series) and SSD SMART connector (ISDCT)

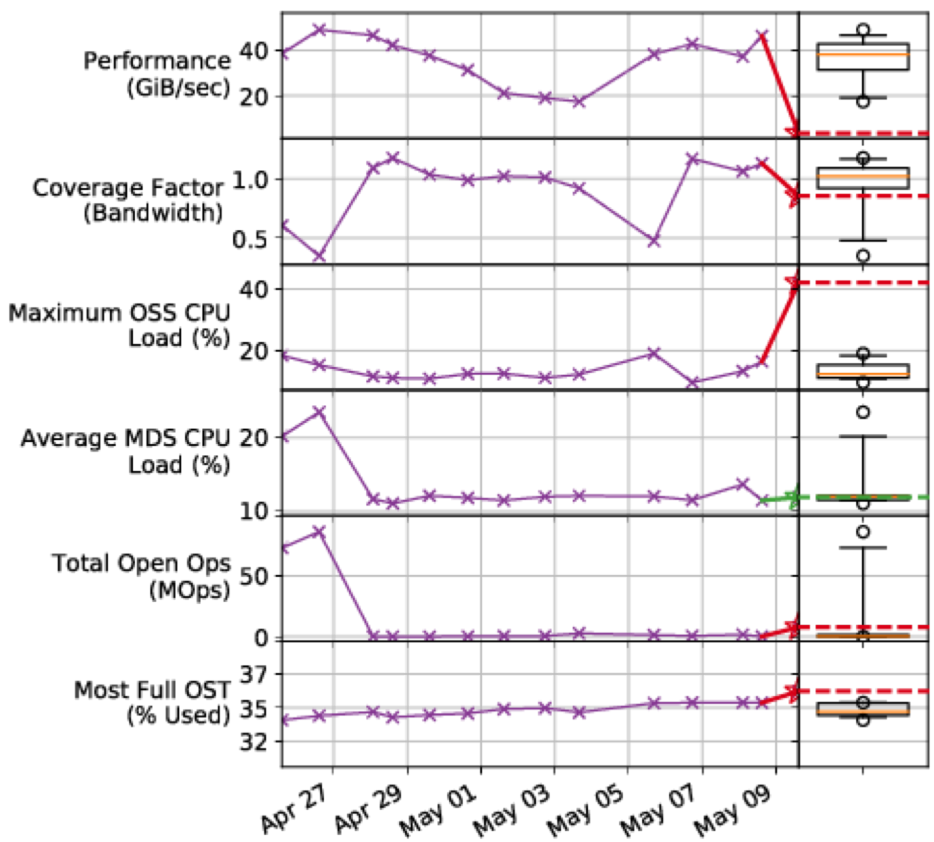
darshan_scoreboard identifies big I/O users

- Index Darshan logs by
  - user
  - file system
  - application name
- SELECT SUM(bytes) WHERE fs = “lustre” GROUP BY user
- 60% of Lustre I/O logged by Darshan came from five users!



All Darshan logs during April 2018 on Cori

What tickles performance?



What metrics actually matter here?

To what degree do they matter?

Use pytokio + scipy!

What tickles performance?

Correlate performance with every other metric over a year's worth of daily benchmark data

	IOR Write	HACC Write	IOR Read	HACC Read
Coverage Factor (Bandwidth)	**+0.3954**	**+0.3944**	**+0.4881**	**+0.3987**
Coverage Factor (opens)	**+0.3006**	**+0.3551**	+0.2034	+0.1503
Average MDS CPU Load	-0.2241	-0.2081	+0.0408	+0.0369
Average OSS CPU Load	**+0.5131**	**+0.3266**	+0.1999	+0.0659
Peak MDS CPU Load	-0.2226	-0.2438	-0.0249	+0.0068
Peak OSS CPU Load	+0.1091	-0.0124	-0.0700	-0.0995
OST Fullness	-0.1834	-0.1553	+0.1211	+0.0697
Number of failed-over OSTs	**-0.4304**	**-0.4209**	-0.1064	-0.0677
Average Job Radius	-0.0140	+0.0301	**+0.3510**	**+0.4061**

Pearson correlation coefficients

Shaded by magnitude, bolded if statistically significant

What tickles performance?

Low data/metadata contention correlates with performance

	IOR Write	HACC Write	IOR Read	HACC Read
Coverage Factor (Bandwidth)	**+0.3954**	**+0.3944**	**+0.4881**	**+0.3987**
Coverage Factor (opens)	**+0.3006**	**+0.3551**	+0.2034	+0.1503
Average MDS CPU Load	-0.2241	-0.2081	+0.0408	+0.0369
Average OSS CPU Load	**+0.5131**	**+0.3266**	+0.1999	+0.0659
Peak MDS CPU Load	-0.2226	-0.2438	-0.0249	+0.0068
Peak OSS CPU Load	+0.1091	-0.0124	-0.0700	-0.0995
OST Fullness	-0.1834	-0.1553	+0.1211	+0.0697
Number of failed-over OSTs	**-0.4304**	**-0.4209**	-0.1064	-0.0677
Average Job Radius	-0.0140	+0.0301	**+0.3510**	**+0.4061**

Pearson correlation coefficients

Shaded by magnitude, bolded if statistically significant

What tickles performance?

Good write performance coincides with high OSS CPU

	IOR Write	HACC Write	IOR Read	HACC Read
Coverage Factor (Bandwidth)	**+0.3954**	**+0.3944**	**+0.4881**	**+0.3987**
Coverage Factor (opens)	**+0.3006**	**+0.3551**	+0.2034	+0.1503
Average MDS CPU Load	-0.2241	0.2081	+0.0408	+0.0369
Average OSS CPU Load	**+0.5131**	**+0.3266**	0.1999	+0.0659
Peak MDS CPU Load	0.2226	-0.2438	-0.0249	+0.0068
Peak OSS CPU Load	+0.1091	-0.0124	-0.0700	-0.0995
OST Fullness	-0.1834	-0.1553	+0.1211	+0.0697
Number of failed-over OSTs	**-0.4304**	**-0.4209**	-0.1064	-0.0677
Average Job Radius	-0.0140	+0.0301	**+0.3510**	**+0.4061**

Pearson correlation coefficients

Shaded by magnitude, bolded if statistically significant

What tickles performance?

Failovers are very bad for writes

	IOR Write	HACC Write	IOR Read	HACC Read
Coverage Factor (Bandwidth)	**+0.3954**	**+0.3944**	**+0.4881**	**+0.3987**
Coverage Factor (opens)	**+0.3006**	**+0.3551**	+0.2034	+0.1503
Average MDS CPU Load	-0.2241	-0.2081	+0.0408	+0.0369
Average OSS CPU Load	**+0.5131**	**+0.3266**	+0.1999	+0.0659
Peak MDS CPU Load	-0.2226	-0.2438	-0.0249	+0.0068
Peak OSS CPU Load	+0.1091	-0.0124	-0.0700	-0.0995
OST Fullness	0.1834	-0.1553	+0.1211	+0.0697
Number of failed-over OSTs	**-0.4304**	**-0.4209**	-0.1064	-0.0677
Average Job Radius	0.0140	+0.0301	**+0.3510**	**+0.4061**

Pearson correlation coefficients

Shaded by magnitude, bolded if statistically significant

What tickles performance?

I/O is sensitive to topology, even on dragonfly!

	IOR Write	HACC Write	IOR Read	HACC Read
Coverage Factor (Bandwidth)	**+0.3954**	**+0.3944**	**+0.4881**	**+0.3987**
Coverage Factor (opens)	**+0.3006**	**+0.3551**	+0.2034	+0.1503
Average MDS CPU Load	-0.2241	-0.2081	+0.0408	+0.0369
Average OSS CPU Load	**+0.5131**	**+0.3266**	+0.1999	+0.0659
Peak MDS CPU Load	-0.2226	-0.2438	-0.0249	+0.0068
Peak OSS CPU Load	+0.1091	-0.0124	-0.0700	-0.0995
OST Fullness	-0.1834	-0.1553	+0.1211	+0.0697
Number of failed-over OSTs	**-0.4304**	**-0.4209**	-0.1064	-0.0677
Average Job Radius	-0.0140	+0.0301	**+0.3510**	**+0.4061**

Pearson correlation coefficients

Shaded by magnitude, bolded if statistically significant

https://github.com/nersc/pytokio

Give pytokio a try!

TOKIO NERSC

You can do all of this on your Cray XC and ClusterStor systems with pytokio *out of the box!*

TOKIO

pytokio: github.com/nersc/pytokio/

TOKIO for ClusterStor: github.com/nersc/tokio-cray/
(available June 2018)

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contracts DE-AC02-05CH11231 and DE-AC02-06CH11357 (Project: A Framework for Holistic I/O Workload Characterization, Program manager: Dr. Lucy Nowell). This research used resources and data generated from 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 and the Argonne Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

U.S. DEPARTMENT OF ENERGY | Office of Science

Argonne NATIONAL LABORATORY

BERKELEY LAB

https://github.com/nersc/pytokio

pytokio: Free bonus offer

Call in the next 30 minutes and also receive…

- **Jupyter notebooks**: demonstrate useful analyses
- **CLI tools**: interact with component-level data
- **Unit tests** (and integration tests, smoke tests, etc): basic usage examples and sample input data sets

https://github.com/nersc/pytokio

pytokio connectors as of May 2018

Connector	Status	Data provided
CraySDB	Implemented	Aries topology information
Darshan	Implemented	Application-level I/O profiling
LMT	Implemented	Lustre server-side monitoring
Slurm	Implemented	Job start/end times, job node lists
Lustre health	Implemented	OST failover, OST fullness
NERSC ISDCT	Implemented	Burst Buffer SMART data
NERSC Jobs DB	Implemented	Job accounting information
NERSC LFS State	Implemented	Lustre OST/MDT fullness and failovers
HDF5/H5LMT	Implemented	Archived LMT and burst buffer server-side data
Collectd/Elasticsearch	Implemented	Server-side data for burst buffer, LNET, DVS, etc
HPSS Summary	Planned	HPSS daily summary data
Spectrum Scale Zimon	Planned	Server-side data for GPFS > 4.2
GPFS ggiostat	Planned	Server-side data for GPFS <= 4.1