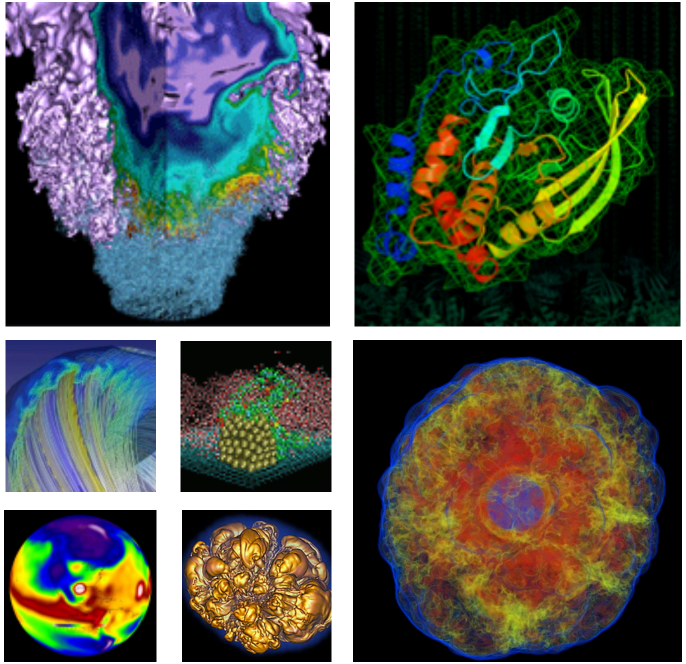
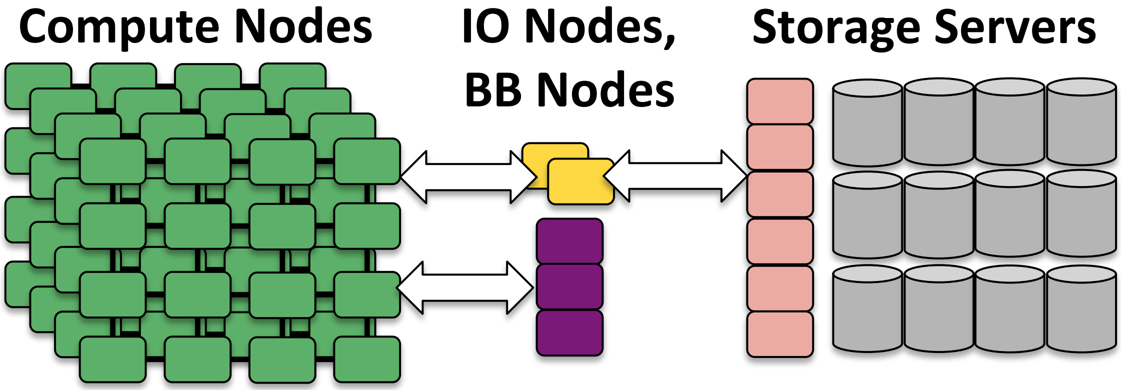
UMAMI: A Recipe for Generating Meaningful Metrics through Holistic I/O Performance Analysis



Glenn K. Lockwood, Shane Snyder, Wucherl Yoo, Kevin Harms, Zachary Nault, Suren Byna, Philip Carns, Nicholas J. Wright

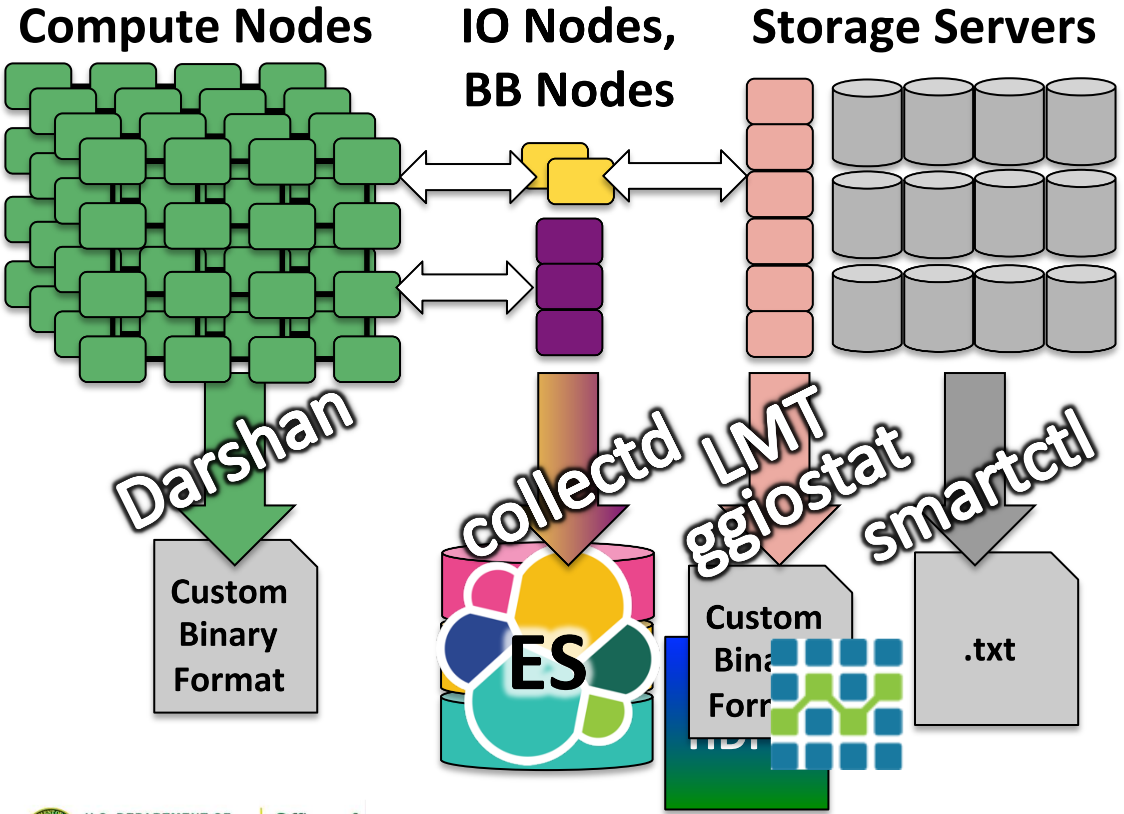
October 27, 2017

Understanding I/O today is hard



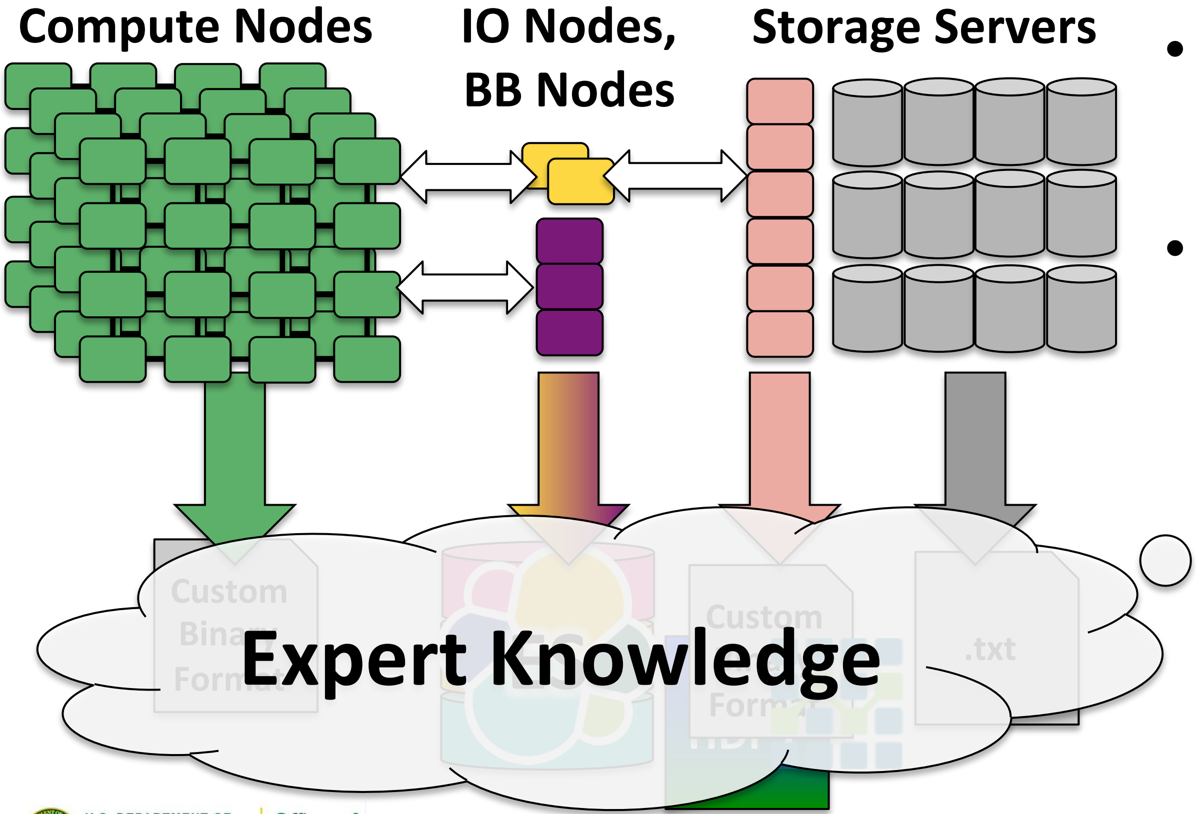
- Storage hierarchy is getting more complicated

Understanding I/O today is hard

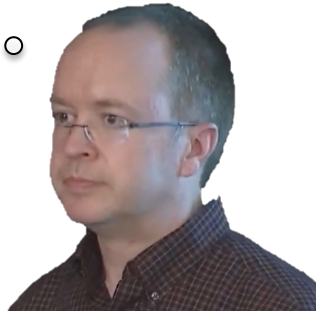


- Storage hierarchy is getting more complicated
- Currently monitor each component separately is standard practice

Understanding I/O today is hard

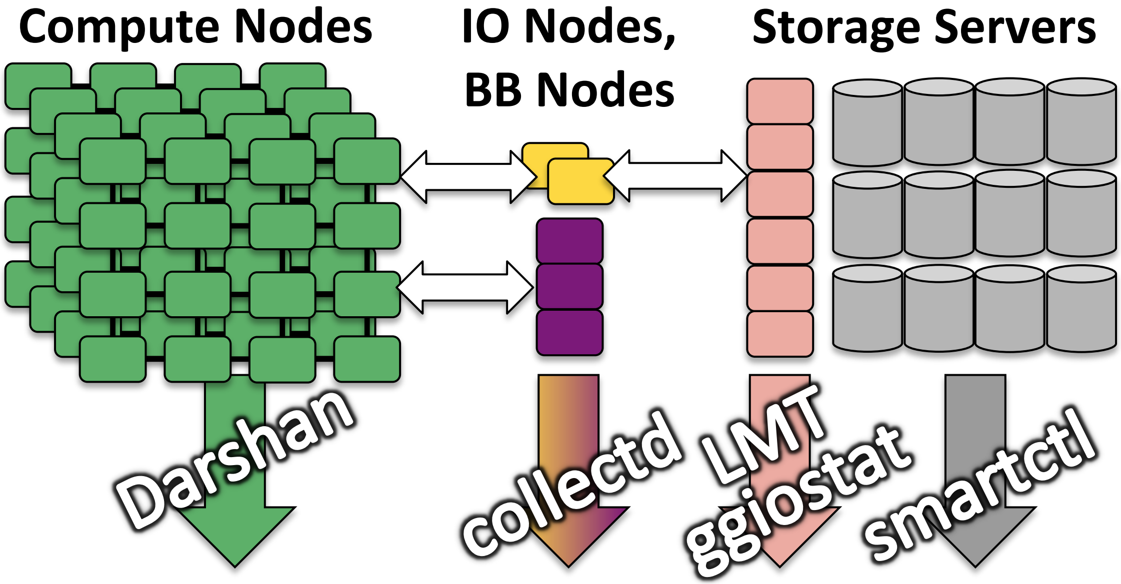


- Storage hierarchy is getting more complicated
- Currently monitor each component separately is standard practice



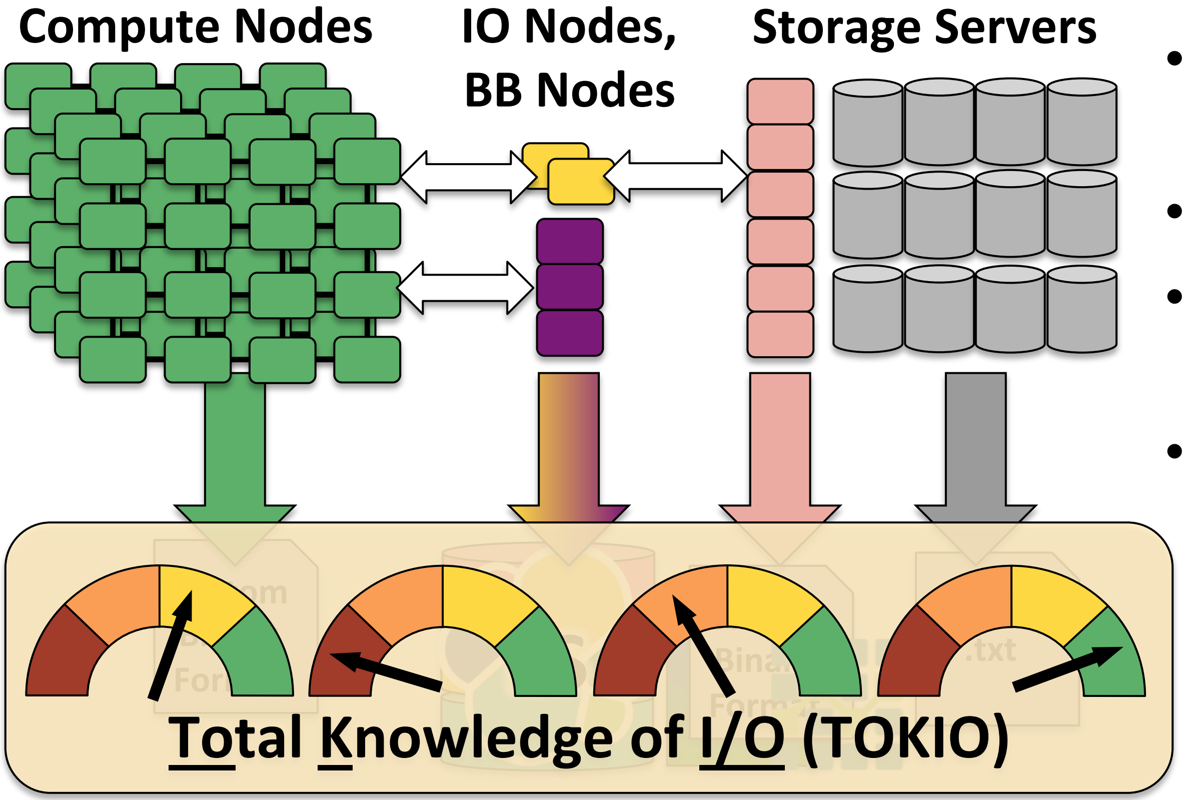
I/O expert (Phil Carns) from ATPESC: https://insidehpc.com/2017/10/hpc-io-computational-scientists/

Total Knowledge of I/O with holistic analysis



- Can we augment expert knowledge?
- Using existing tools?

Total Knowledge of I/O with holistic analysis

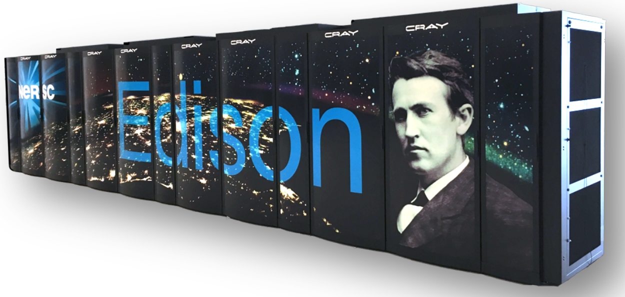


- Can we augment expert knowledge?
- Using existing tools?
- Combine, index, and normalize their metrics
- Provide a holistic view

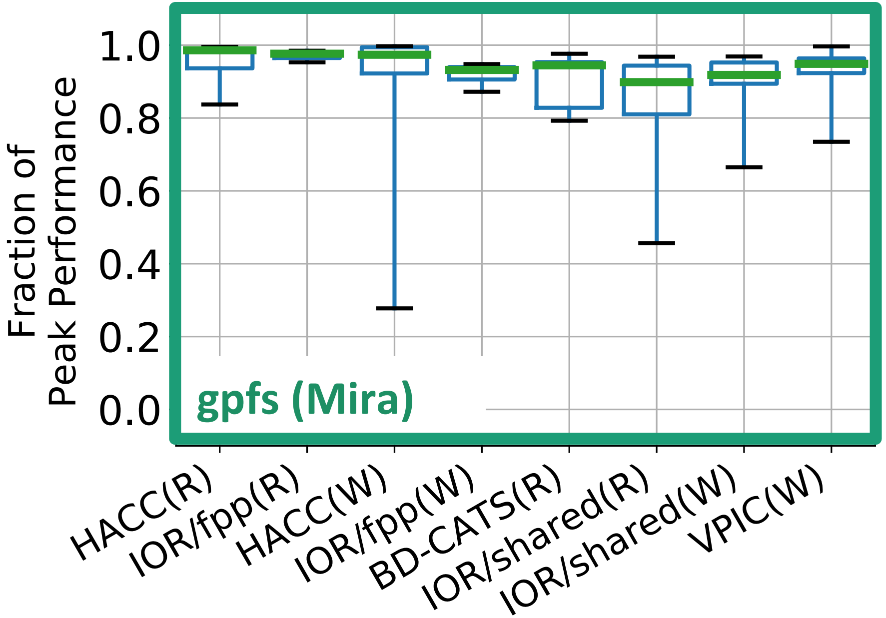
I/O expert (Phil Carns) from ATPESC: https://insidehpc.com/2017/10/hpc-io-computational-scientists/

What is possible with holistic I/O analysis?

- **Run four different I/O workloads every day for a month**
  - Jobs scaled to achieve > 80% of peak fs performance
  - Exercise file-per-proc, shared file, big and small xfers
- **Run on ALCF Mira (IBM BG/Q) and NERSC Edison (Cray XC)**
  - One GPFS file system on Mira (gpfs-mira)
  - Two Lustre file systems on Edison (lustre-reg and lustre-bigio)
- **Use data from production monitoring tools at ALCF and NERSC**
  - Darshan for application-level I/O profiling
  - GPFS and Lustre-specific server-side monitoring tools



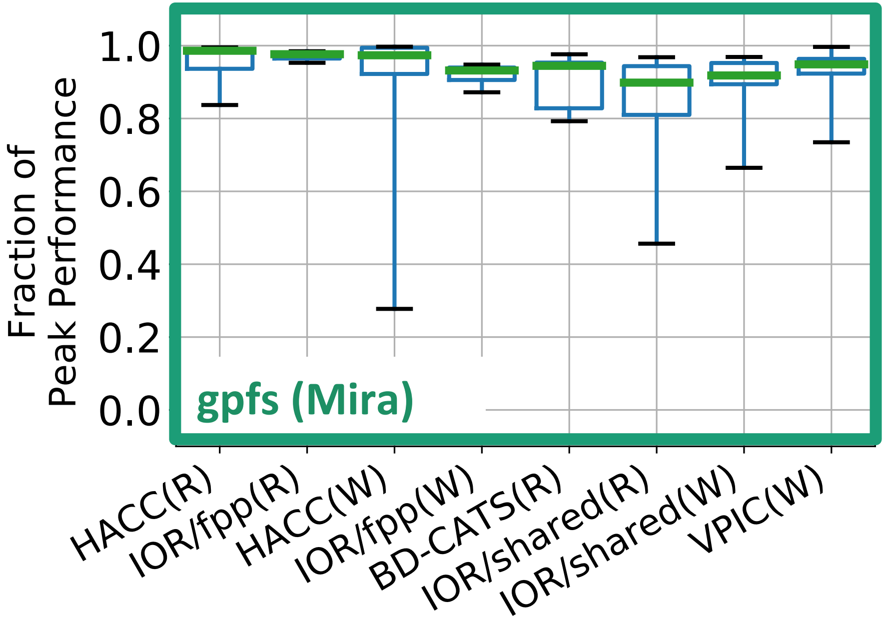
Defining performance variation



- "*Fraction of Peak Performance*" is relative to max performance *for that app on that file system*
- Normalizes out the effects of application I/O patterns and peak file system performance

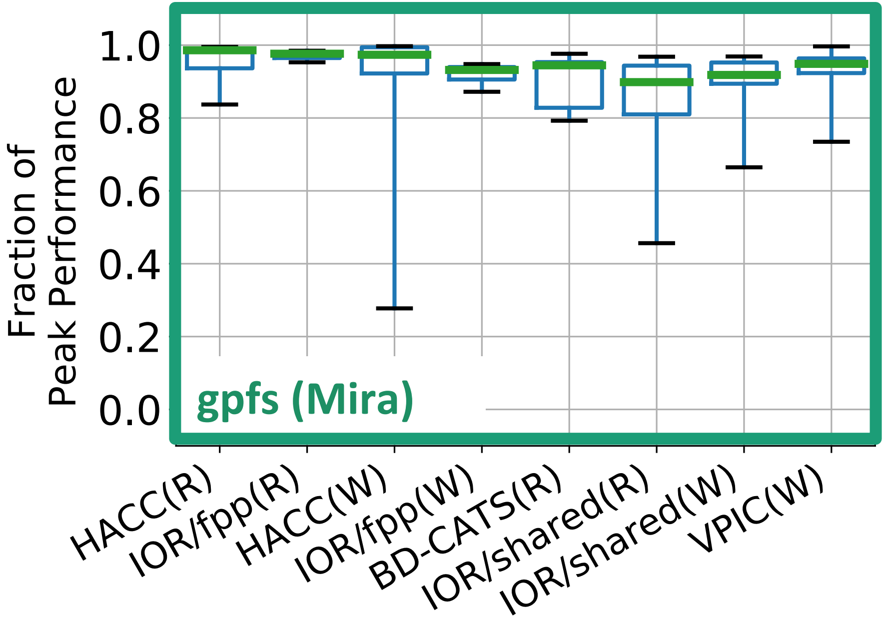
U.S. DEPARTMENT OF ENERGY | Office of Science
Argonne NATIONAL LABORATORY
BERKELEY LAB

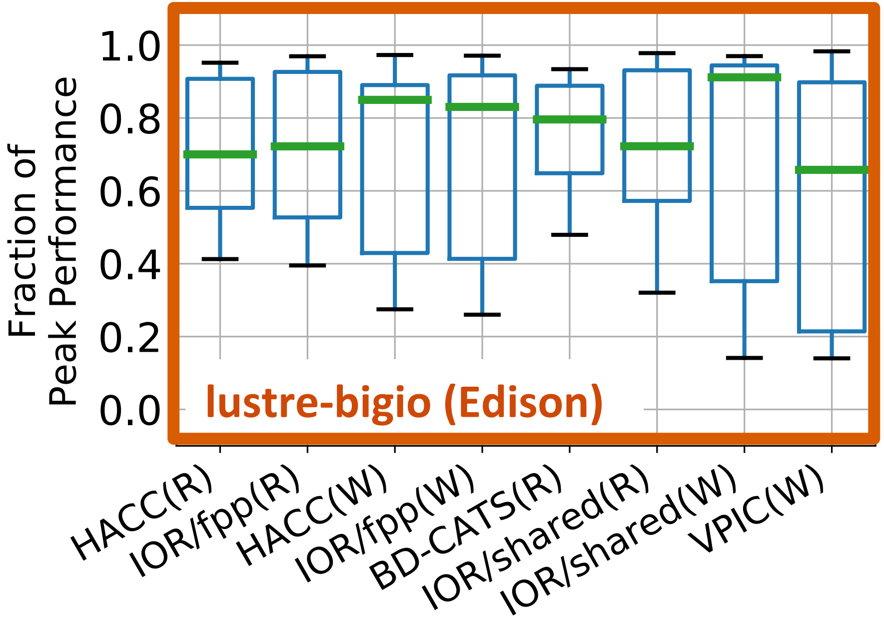
Variation due to application I/O pattern



- "Bad I/O patterns" can cause
  - bad performance
  - bad performance ***variation***
- Some application patterns are more susceptible to high amounts of variation!

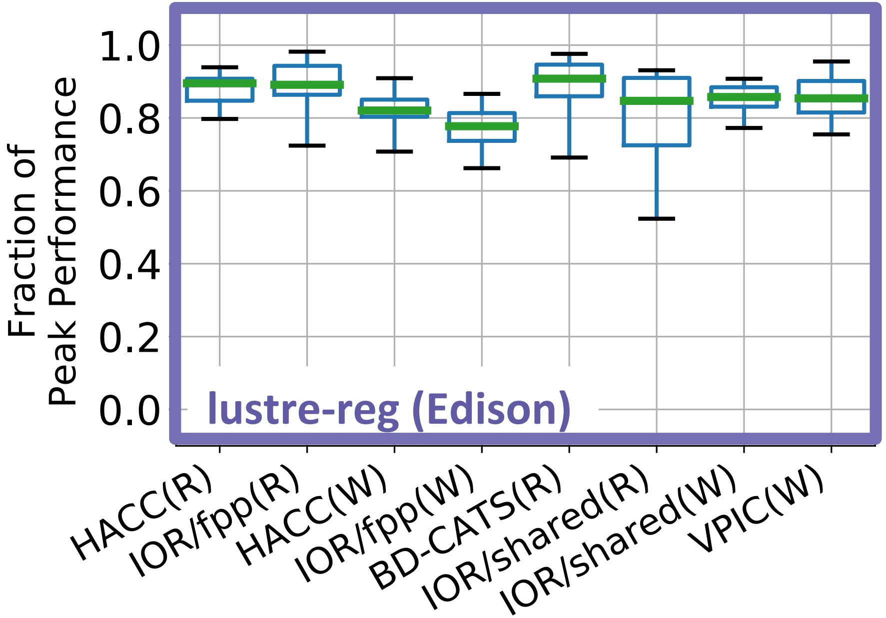
Variation across file system architectures

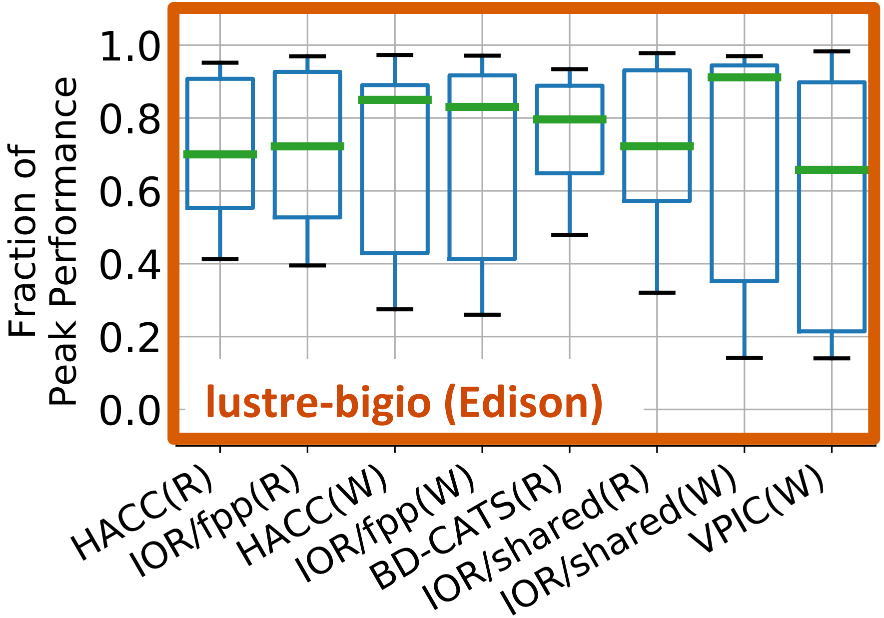




Application I/O patterns are not the only contributor to performance variation

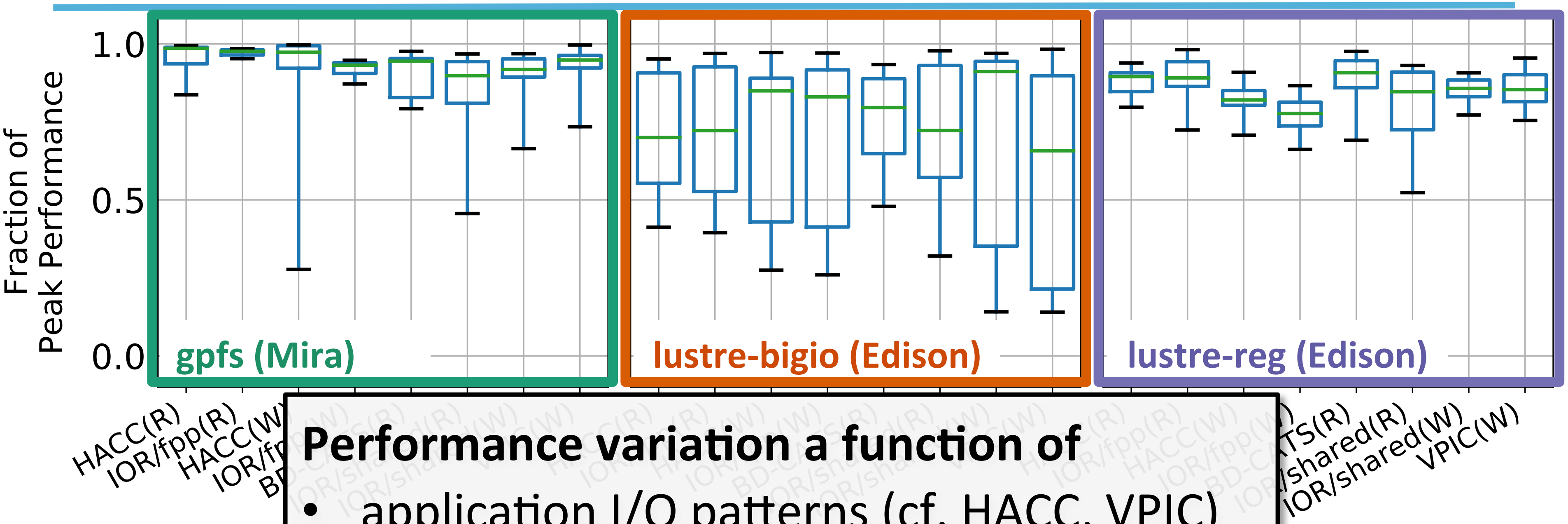
Variation between Lustre configurations





Significant differences even on similar Lustre file systems—other factors (configuration, workload) also matter!

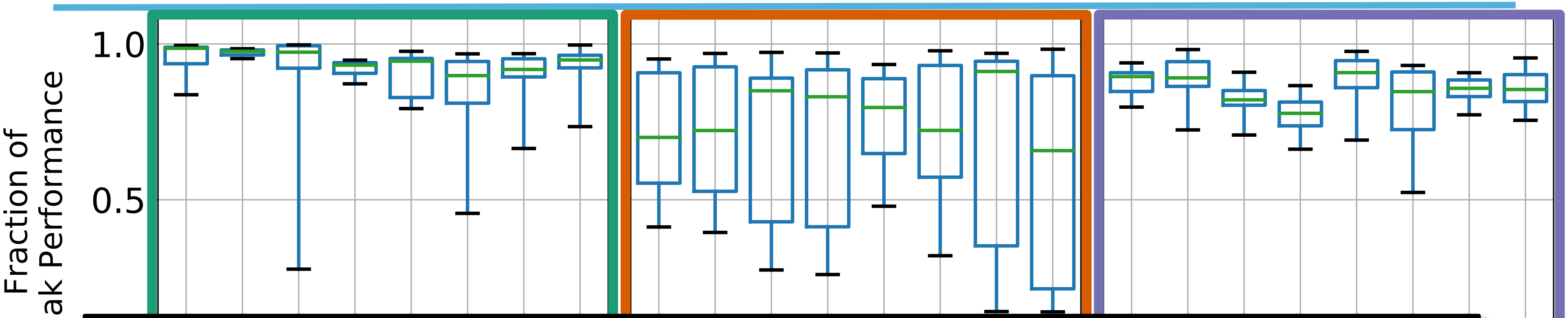
What does this tell us about variation?



Performance variation a function of

- application I/O patterns (cf. HACC, VPIC)
- architecture (cf. gpfs, lustre-bigio)
- other factors (cf. lustre-bigio, lustre-reg)

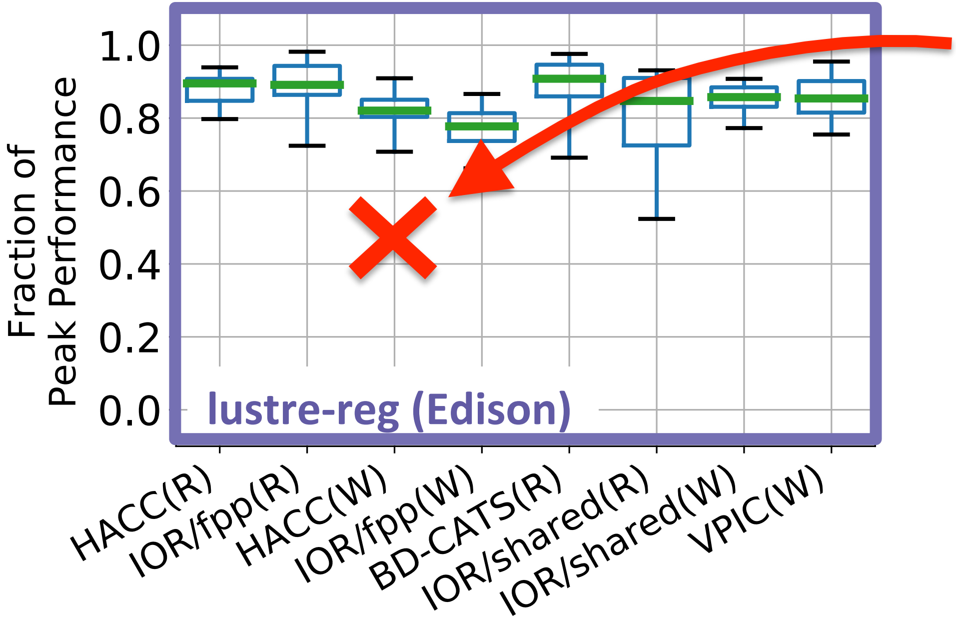
What does this tell us about variation?



File systems have their own "I/O climate"
(like Berkeley vs. Argonne)

Understanding these "other factors" (climate) holistically is essential to understanding performance variability!

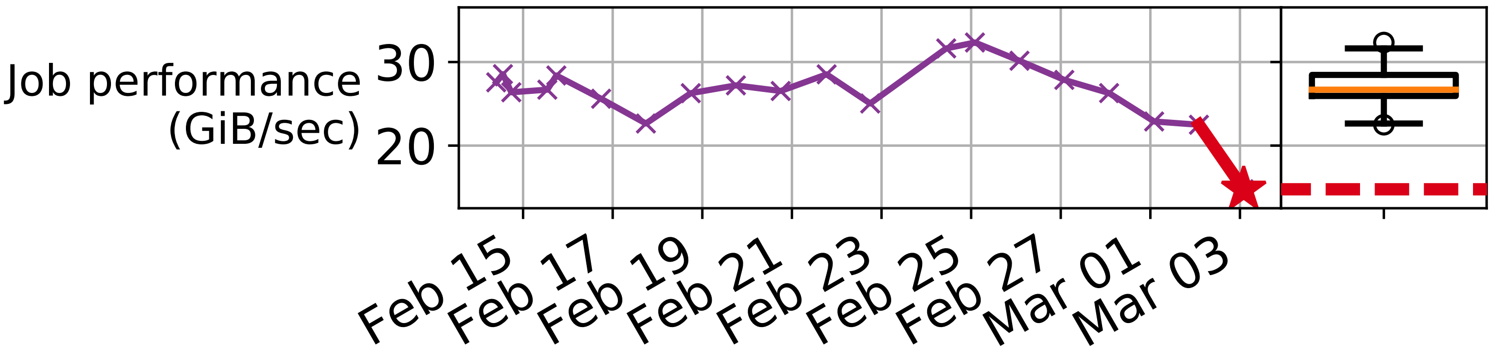
Exploring I/O weather and climate



Let's look at a few cases of bad performance using a **Unified Monitoring and Metrics Interface (UMAMI)**

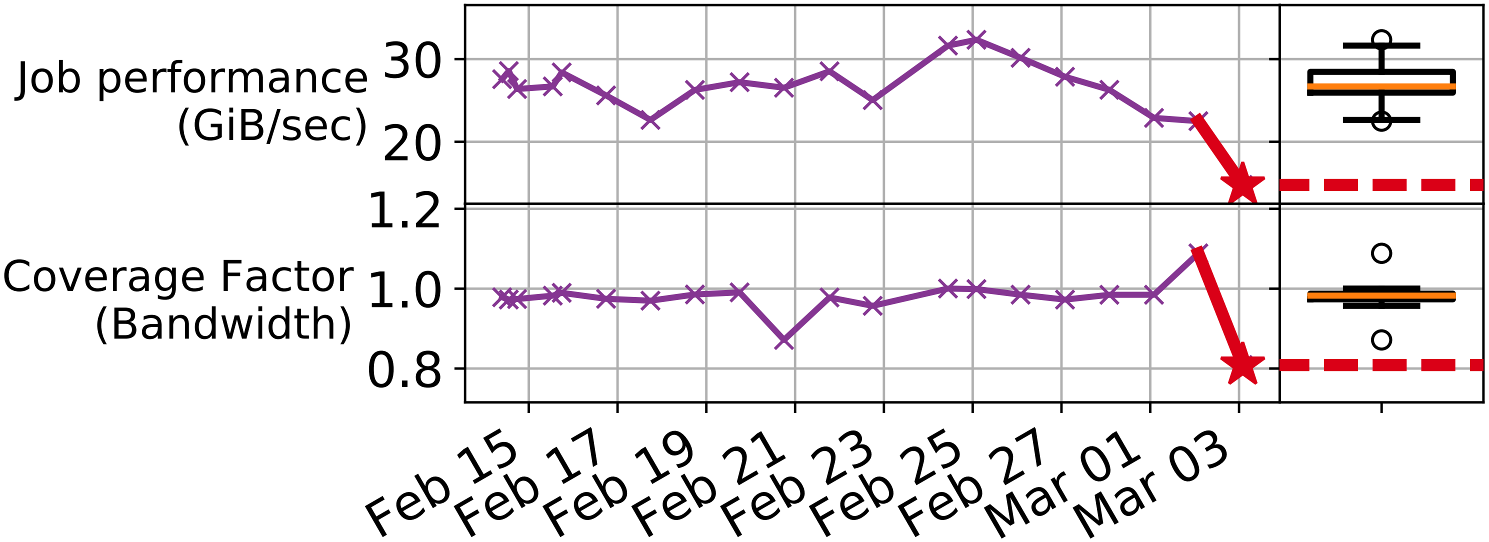
What can a holistic view (climate) tell us about performance (weather)?

Case Study #1: HACC write performance on lustre-reg



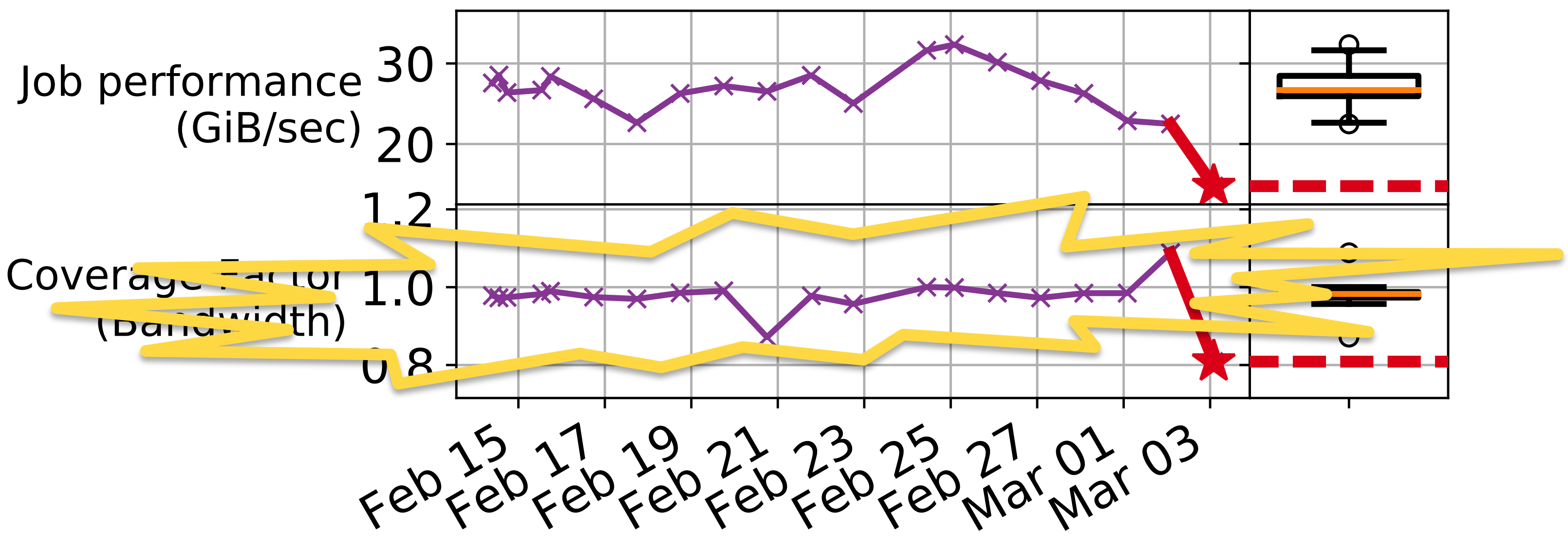
- Is this a snowy day at Argonne or a snowy day at Berkeley?
- Quantitatively define "bad" based on quartiles
- Use UMAMI to determine which aspects of weather were "bad"

Case Study #1:
First guess: blame someone else



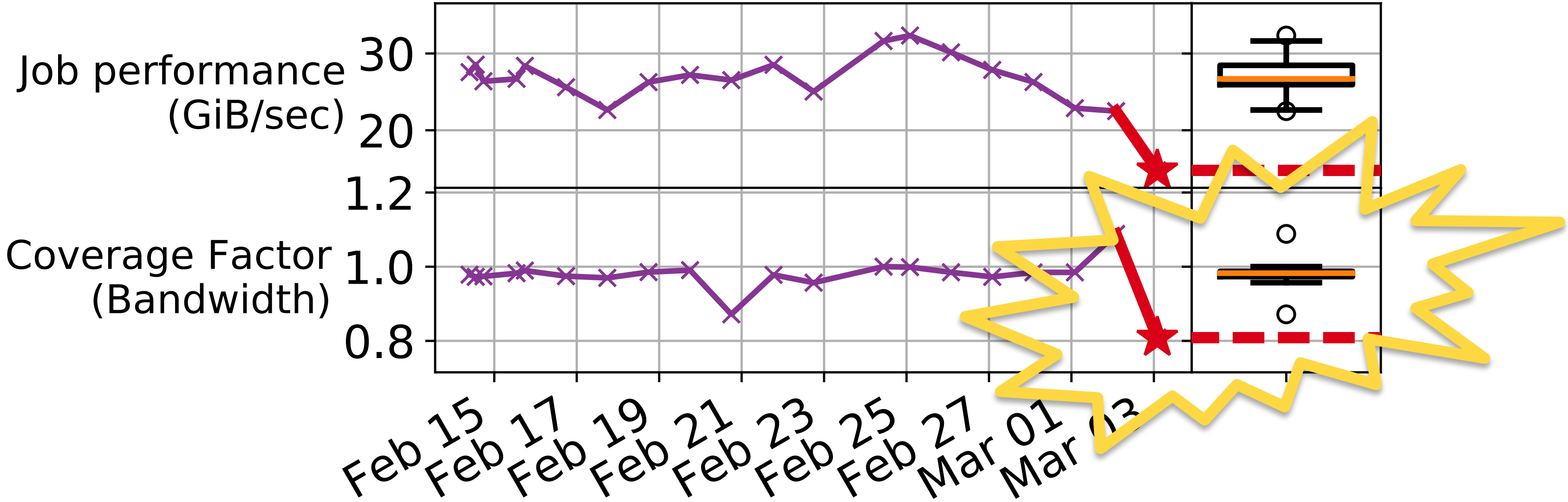
Coverage Factor = how much global bandwidth was consumed by my job?

Case Study #1:
Add *Coverage Factor* to UMAMI



Most jobs get exclusive access to Lustre bandwidth
$(CF_{bw} \approx 1.0)$

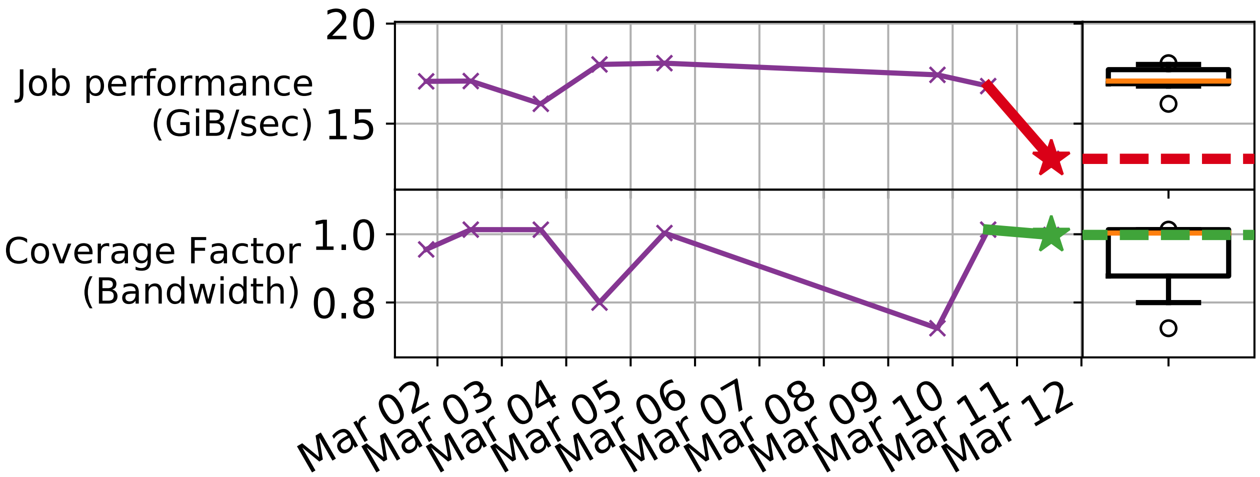
Case Study #1:
Add *Coverage Factor* to UMAMI



Bad performance coincided with low CF

Performance variation caused by bandwidth contention

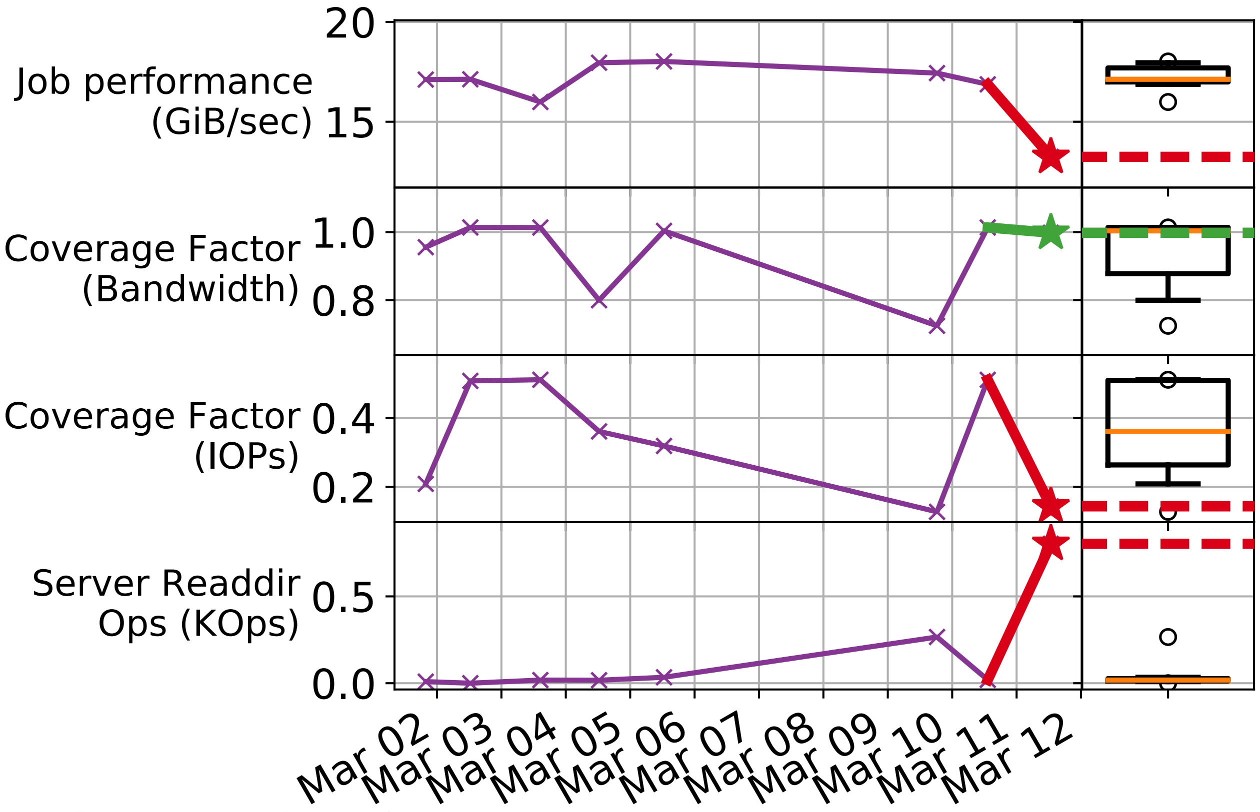
Case Study #2:
VPIC/GPFS: when bandwidth contention isn't the issue



Bad performance *did not* coincide with low CF

Either use expert knowledge or statistical analysis to add more metrics

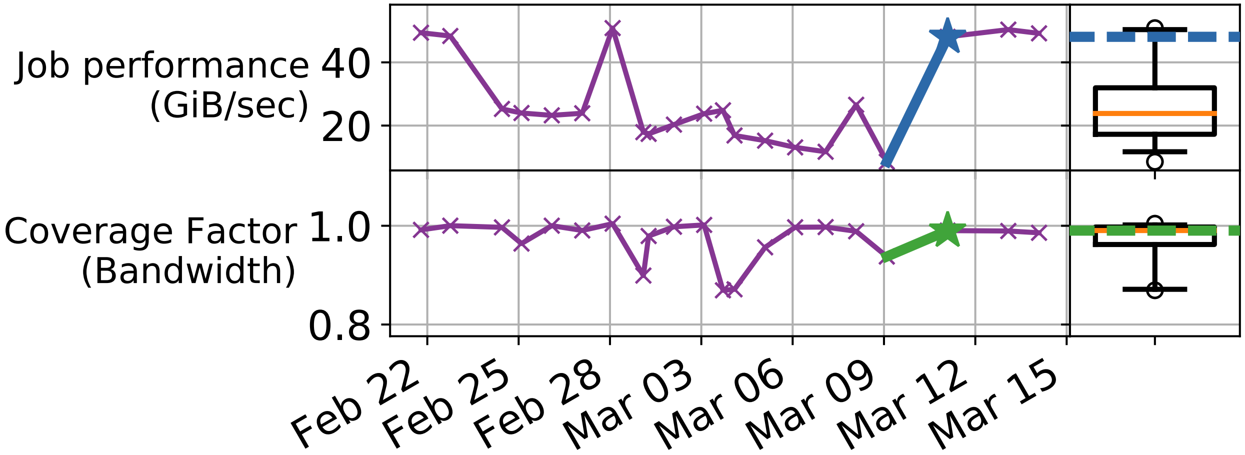
Case Study #2:
VPIC/GPFS: when bandwidth contention isn't the issue



Statistically "bad" levels of contention for metadata IOPS

Performance loss affected by file system implementation

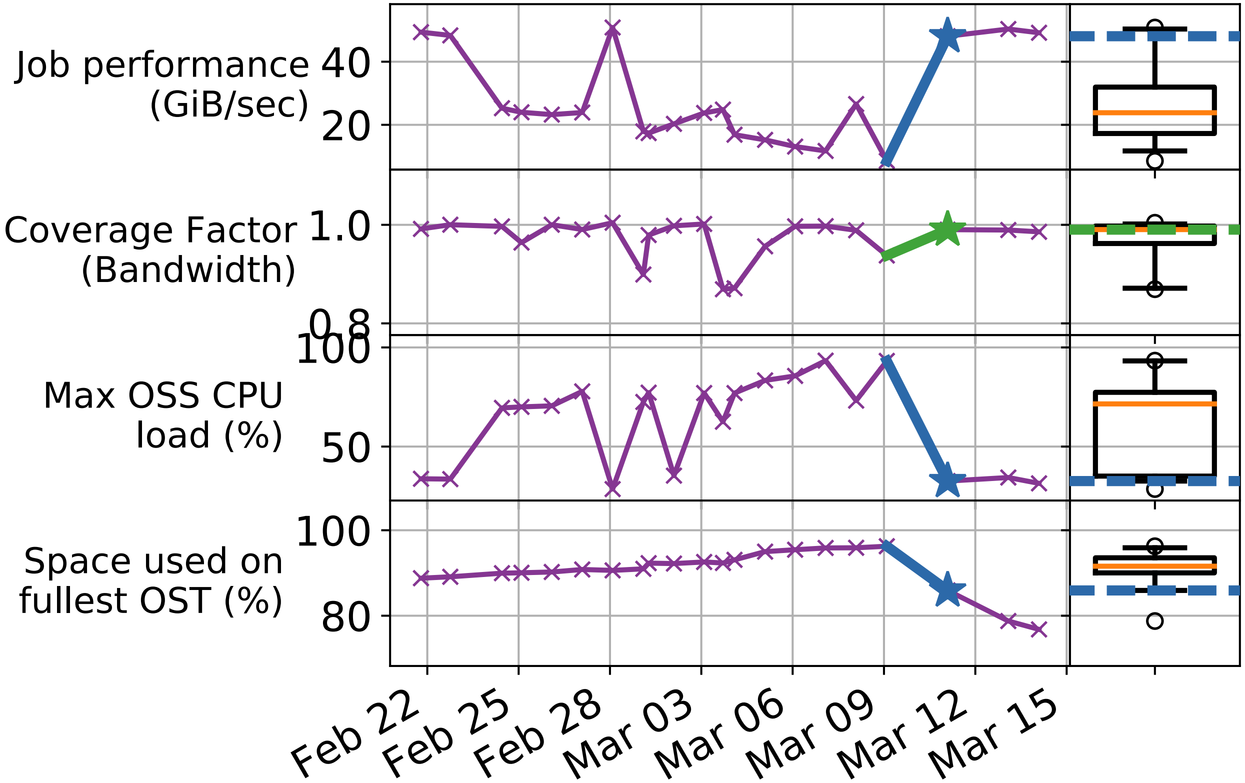
Case Study #3:
HACC/lustre-bigio: effects of "I/O climate change"



Abnormally good performance revealed a long-term bad I/O climate

Bandwidth contention was not the culprit

Case Study #3: HACC/lustre-bigio: effects of "I/O climate change"

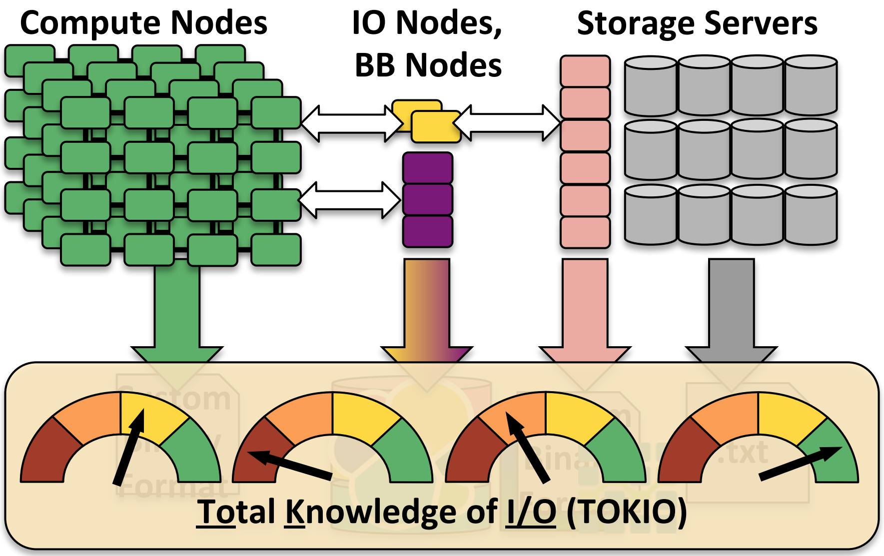


- Moderate negative correlation with OSS CPU load
- Strong negative correlation with file system fullness
- Result of Lustre block allocation at >90% fullness

Conclusions

- **Performance variability is a function of <u>file system climate</u>:**
  - file system architecture
  - overall system workload
  - file system configuration (default striping, etc) and health
- **No single metric predicts variation universally; many factors *can* affect <u>I/O weather</u>:**
  - bandwidth contention
  - metadata op contention (GPFS)
  - file system fullness (Lustre)
- **A holistic view of the storage subsystem is essential to understand performance on complex I/O architectures**

Closer to Total Knowledge



- **Incorporate machine learning**
  – Cluster similar I/O motifs to define I/O climates
  – Infer critical metrics to remove expert from the loop
- **Join the TOKIO effort!**
  – Open source & development – contributions welcome!
  – https://github.com/nersc/pytokio/
  – Support for new component-level tools being added regularly

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.