SAND2018-10543C

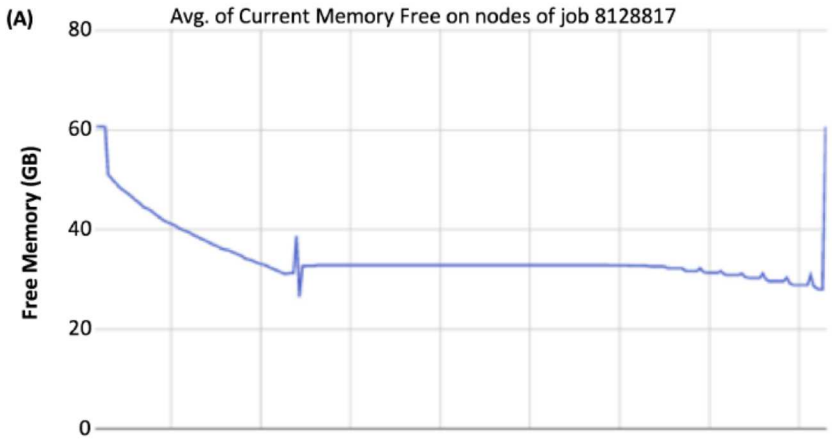
Large-Scale System Monitoring Experiences and Recommendations

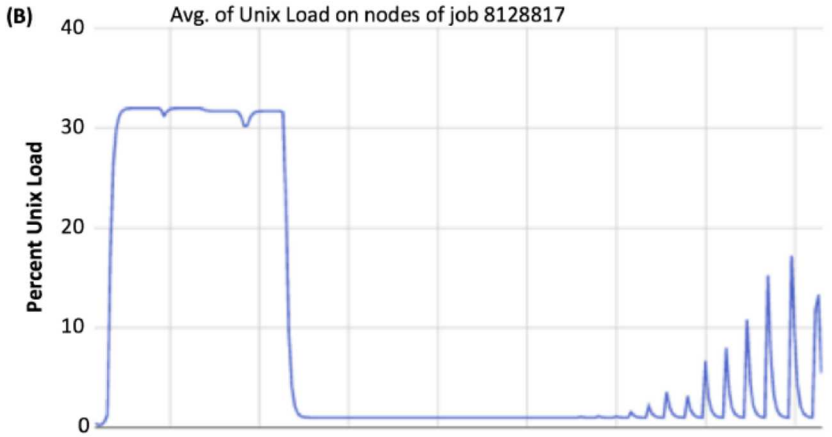
Outline

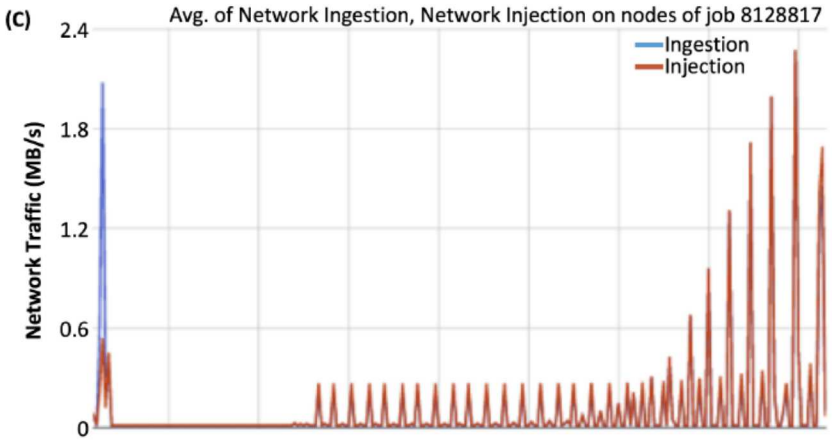
- Motivation for Monitoring at **Ten** Large-scale HPC sites
- State of the Practice Examples
- Spectrum of Approaches
- Needs and Requirements for Comprehensive Monitoring
- Conclusions

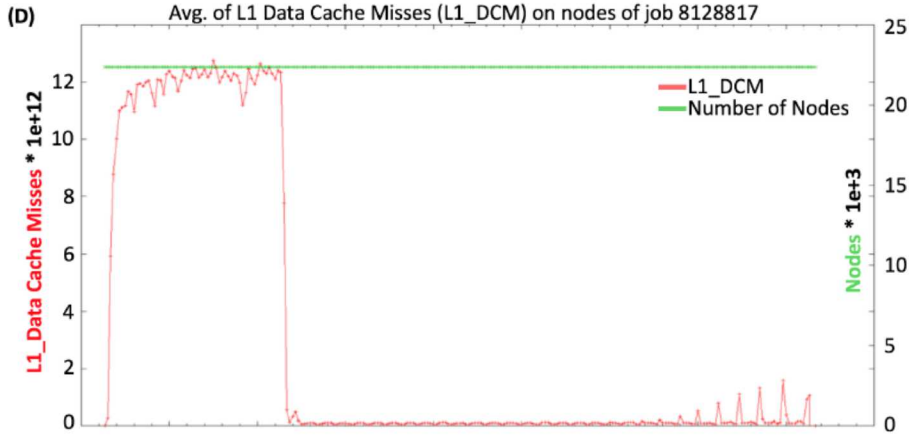
Motivation for Monitoring

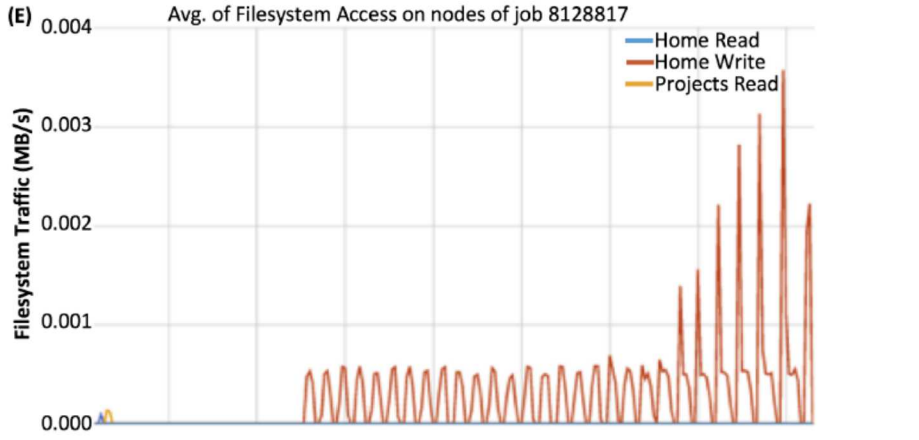
- Monitoring is low-priority in machine acquisitions
  - Emphasis on potential component and sub-system (e.g., network, filesystem) performance
- However, performance understanding, resource utilization, and problem detection and diagnosis are key to overall performance and accurate acquisition requirements

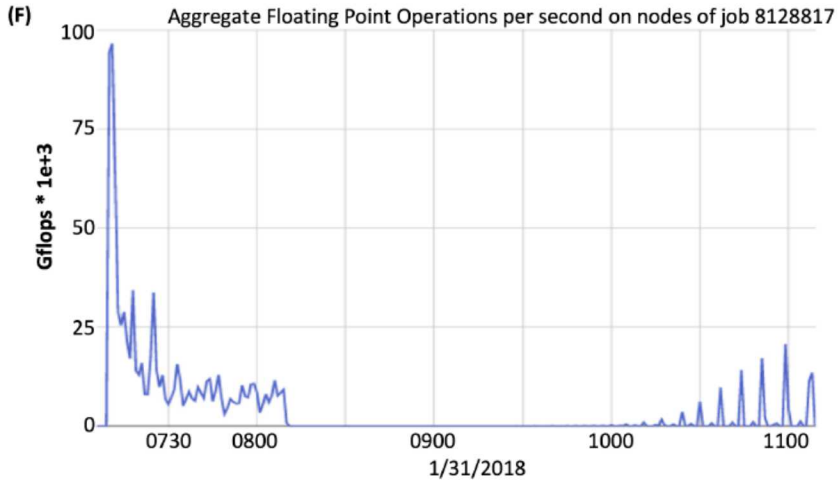












System level monitoring can provide application performance insights

- Job is not doing much for a long stretch of runtime
- Algorithm-matrix size mismatch
- *Reducing* the allocation size request results in better performance

State of the Practice

- Asked ten large-scale HPC sites for success stories in monitoring:
  - Continuous Testing: LANL, NCSA, NERSC
  - GPU: CSCS, ORNL
  - Power: KAUST, SNL
  - MCDRAM: ACLF
  - HSN: HLRS, SNL
  - Environment: NERSC
  - Trend Analysis: ALCF
  - System Utilization and Queue Length: CSC, NERSC
  - Job Analysis and Reporting: NCSA
- While all appear to be site-specific approaches to site-specific problems, all seek to detect and diagnosis sub-optimal operations

Outline

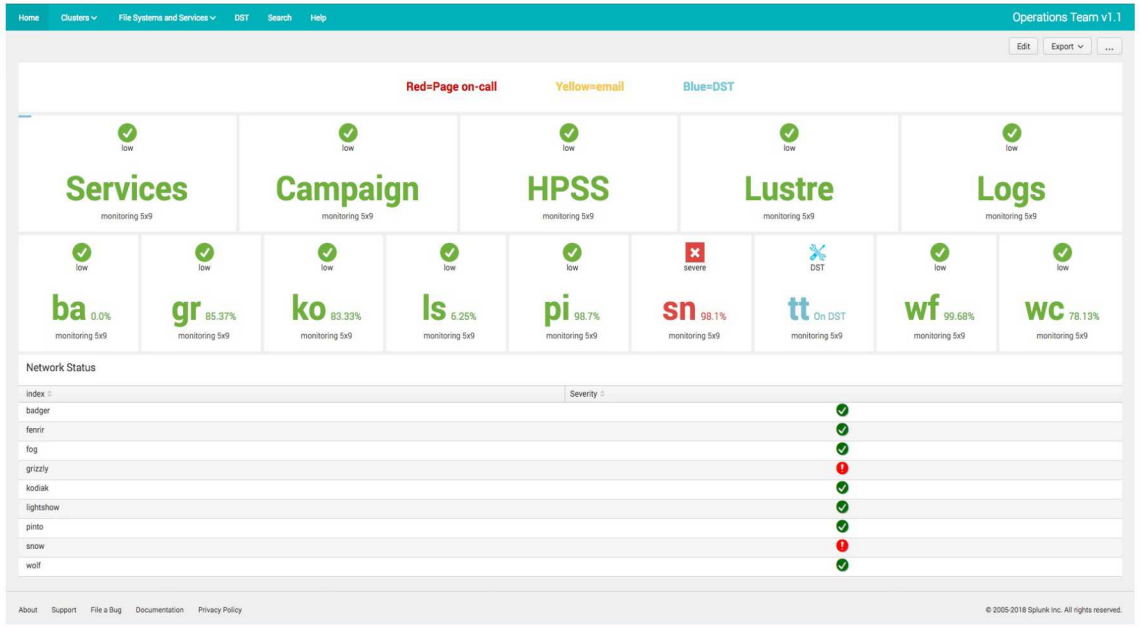
- Motivation for Monitoring at **Ten** Large-scale HPC sites
- State of the Practice Examples
- Spectrum of Approaches
- Needs and Requirements for Comprehensive Monitoring
- Conclusions

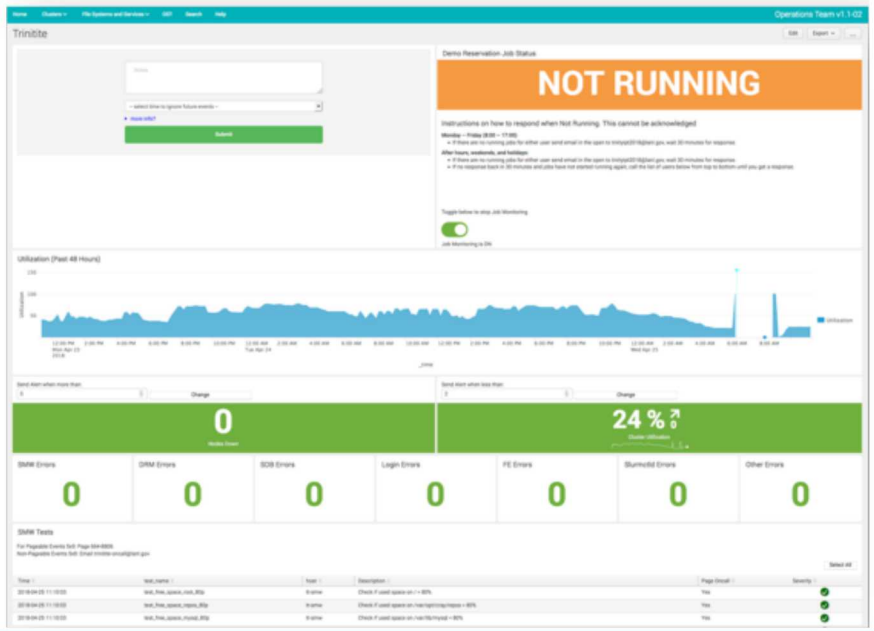
LANL Trinity 20K node XC

Goal: Early detection of problems and anomalies

Approach:

- Developed 70+ health and performance tests run at 10 min intervals (e.g, configurations, verification of essential services and daemons, ensuring free memory on compute nodes and space in shared filesystems.
- Results dashboard includes log data occurrences to facilitate problem diagnosis.

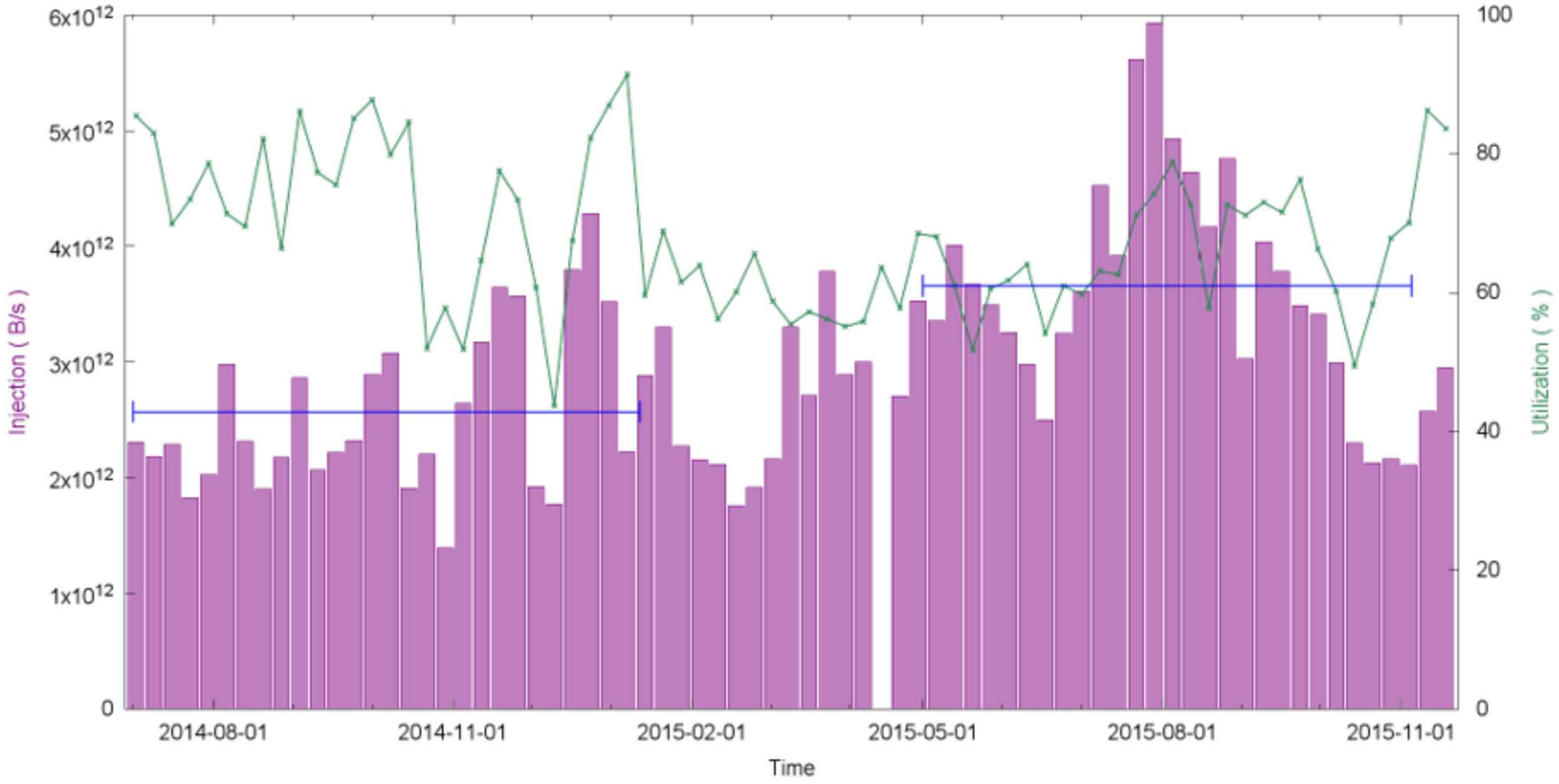




NCSA Blue Waters 27648 node XE/XK

Goal: Facilitate efficient use of resources, early detection of poor performance in critical shared resources, comparison of performance metrics over time, minimize troubleshooting time to solution

Approach: Collect and assess performance and utilization data from all major components and subsystems at one minute intervals



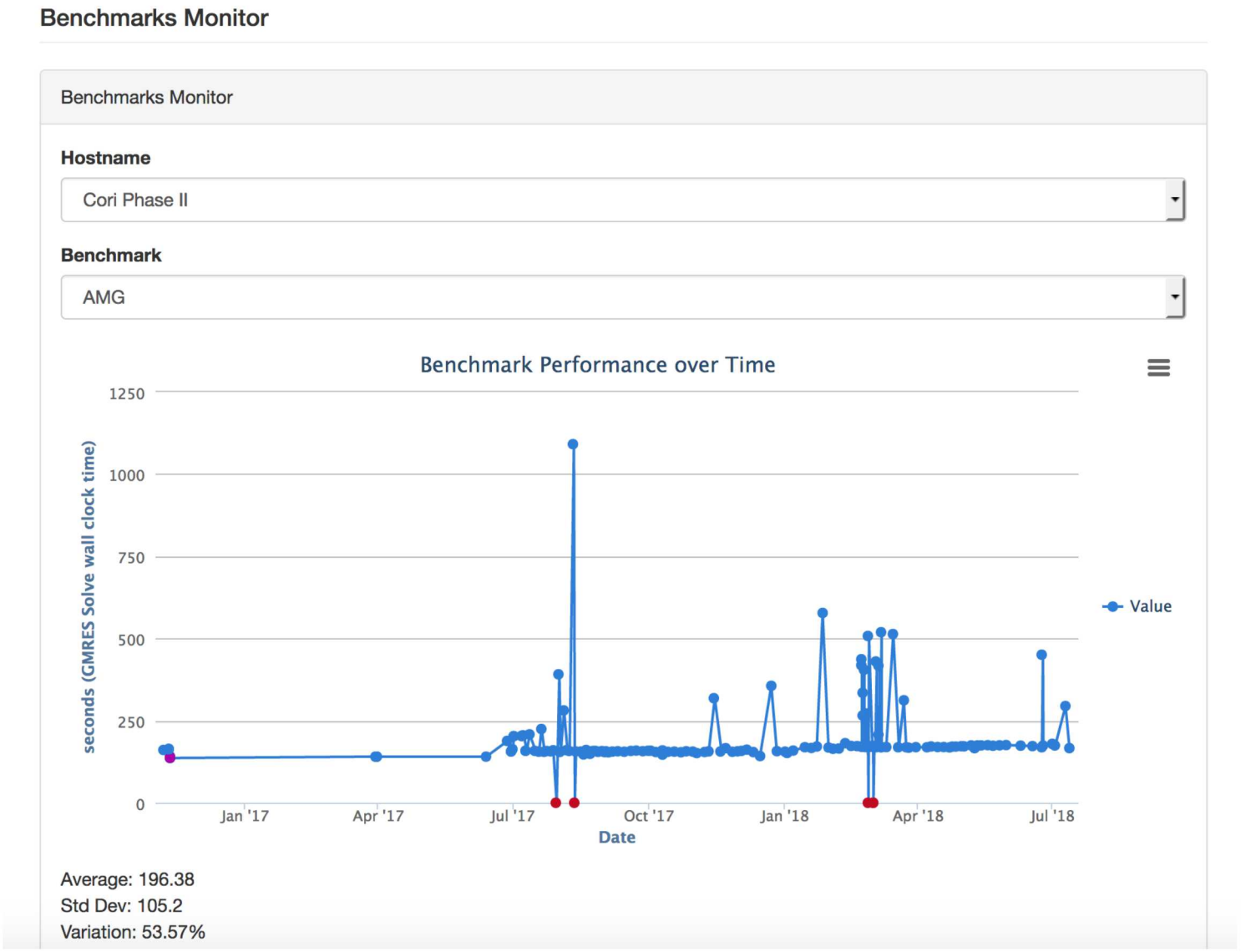
Comparison of resource utilization (injection into the network) under different scheduling and resource allocation strategies. network resources utilized more after TAS (right) than before (left). Blue line represents mean bandwidth utilization as a percent of max.

NERSC Edison 5,586-node, Cori 12,076-node XC

Goal: Identify abnormal or unsatisfactory behaviors

Approach: Run a suite of custom benchmarks that exercise compute, network, and I/O functionality, and publishes performance over time on user-facing web pages

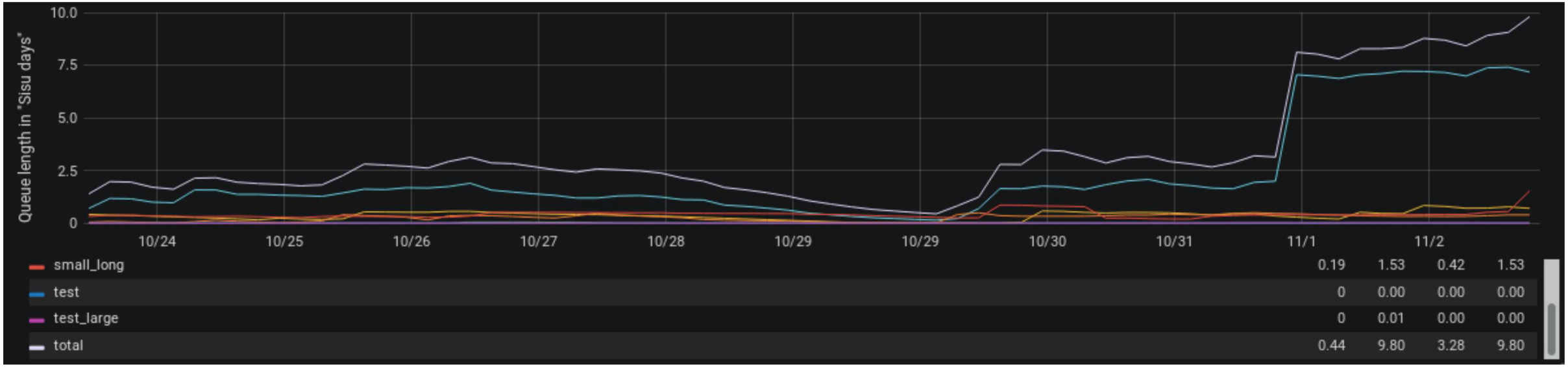
Occurrences and onset of performance problems are apparent in visualizations tracking performance over time.



CSC Sisu 1,688 node XC

Goal: Provide users a realistic view into the expected wait time which may reveal application or component problems

Approach: Calculate the total CPU time scheduled for execution and present it in easy to understand units in a flexible graphical user interface



Queue length development presented in a Graphana view

CSCS Piz Daint 5,320 XC50 nodes w/ NVIDIA GPUs and 1,813 XC40 nodes w/o GPUs

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Goal: Validate health of the GPUs

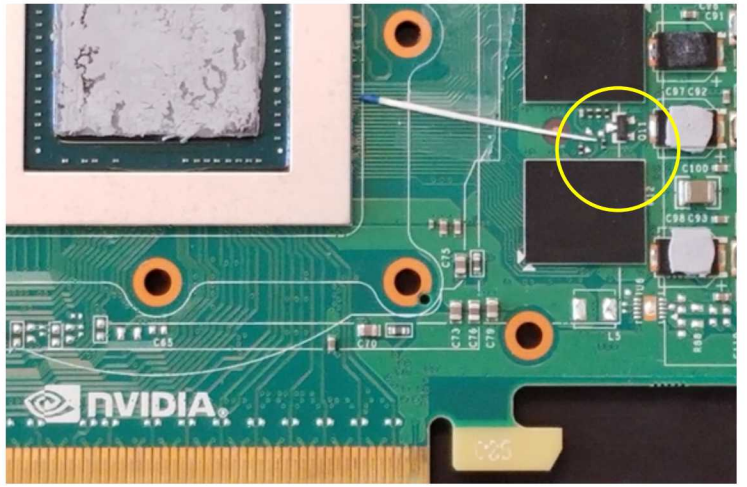
Approach: Developed test suite run integrated into prolog and epilog. Includes automatic removal of components failing tests from service.

ORNL Titan 18,688 node XK7

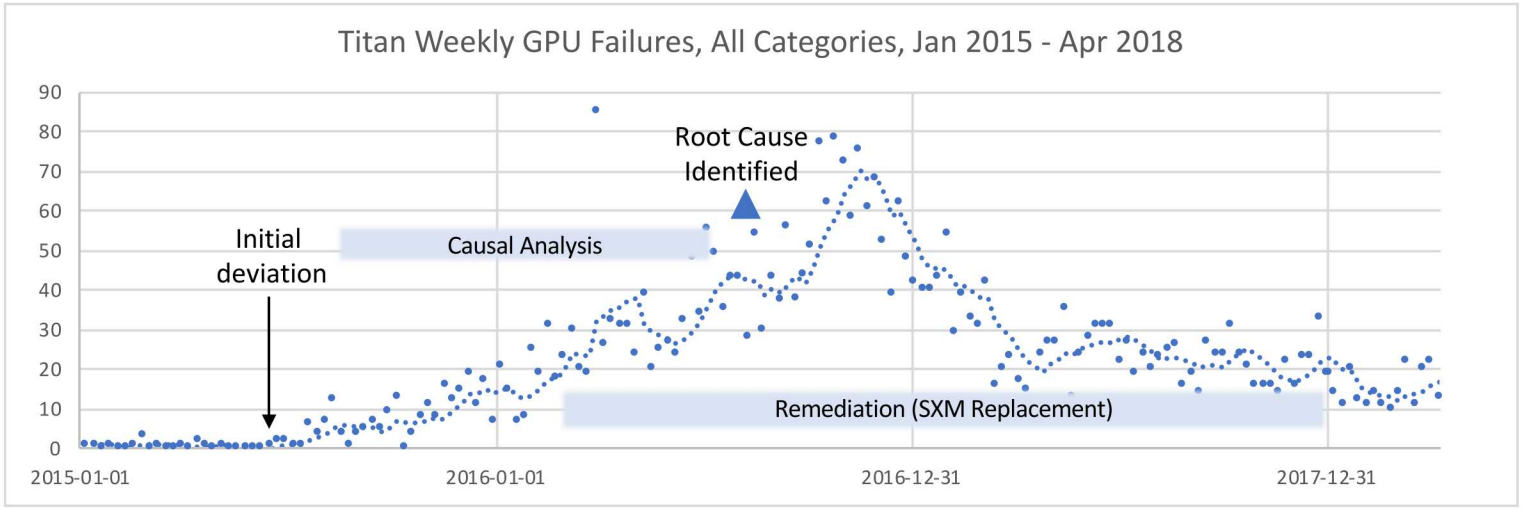
OAK RIDGE National Laboratory

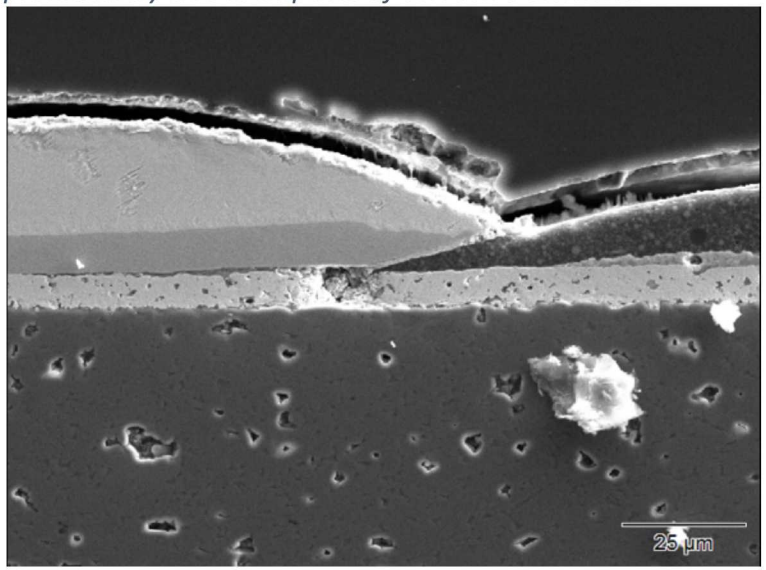
Goal: Assess and increase reliability

Approach: Tracked failure rates. Diagnostic analysis enabled root cause (manufacturing problem). Reduce potential problems through environmental monitoring and enforcing additional standards in procurements.



NVIDIA SXM – Location of a non-ASR that could fail prematurely due to impacts of corrosive elements





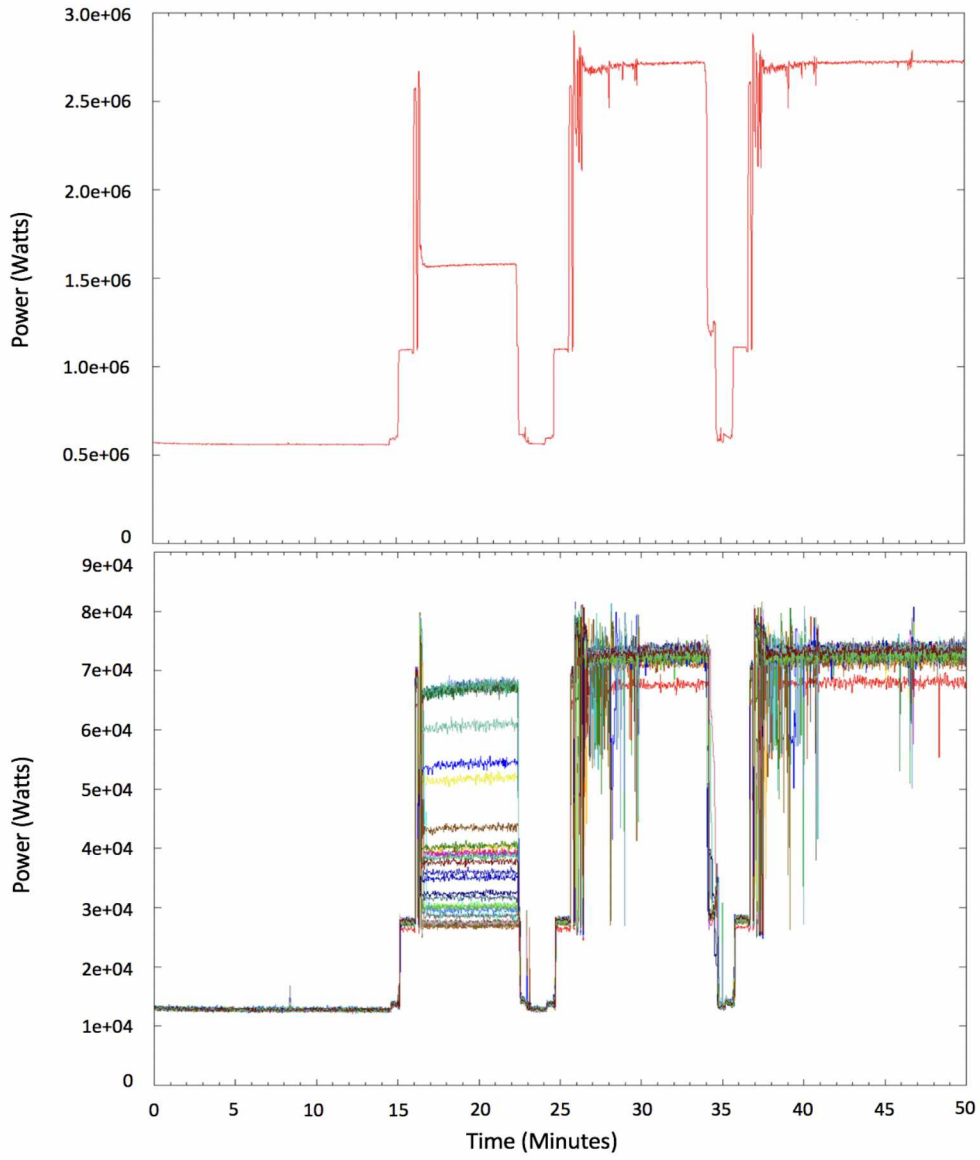
A non-ASR resistor cross-section demonstrating failure due to the formation of silver-sulfide

KAUST Shaheen2 6,174 node XC40

King Abdullah University of Science and Technology

Goal: Stay within power budget, Discover poor performance in production

Approach: Build system and application power profiles. Examine live data for imbalance and irregularity

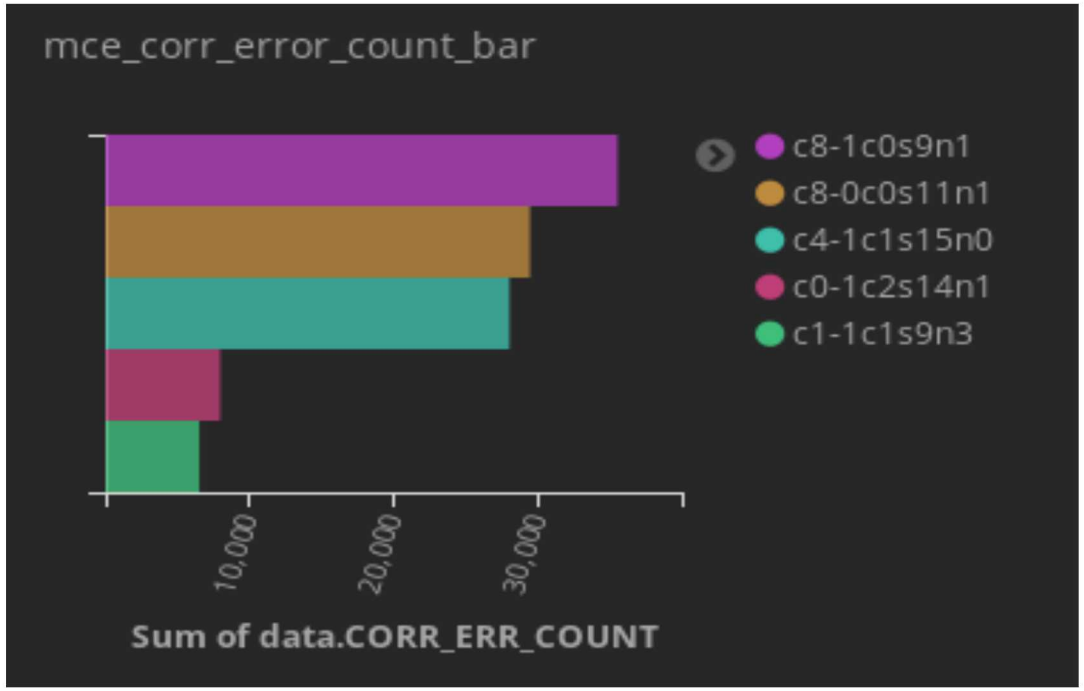


Load imbalance issue was detected with power usage variation. Overall power usage (top); power usage per cabinet (bottom). Around 17-22 minutes, power usage variation of up to 3 times was observed between different cabinets and full system power draw was almost 1.9 times lower during this period of variable cabinet usage. This was rectified for subsequent runs.

ANL Theta 4,392 node XC40

Goal: Gain critical understanding of system state

Approach: Facilitate analysis by writing a tool to obtain data from Cray's event stream (ERD) (as opposed to log files) and make it directly available for analysis

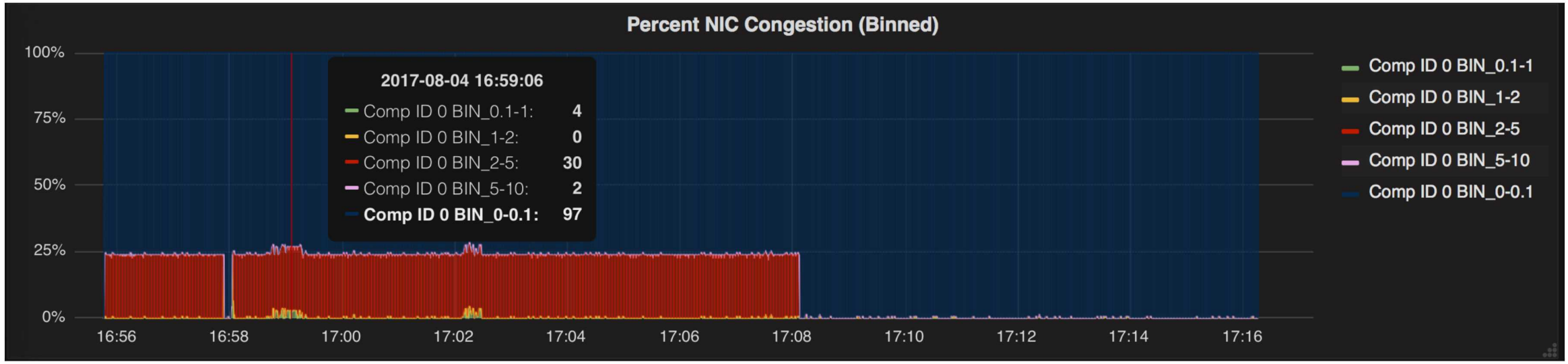


A 24-hour bar graph showing total correctable error counts from compute nodes

SNL Trinity and testbed Mutrino 100 node XC40

Goal: Determine congestion levels, congestion regions, and impact on application performance.

Approach: Collect whole system network performance counters at intervals O(sec). Run-time computation and dashboard of potential congestion measures (collab. with Cray).

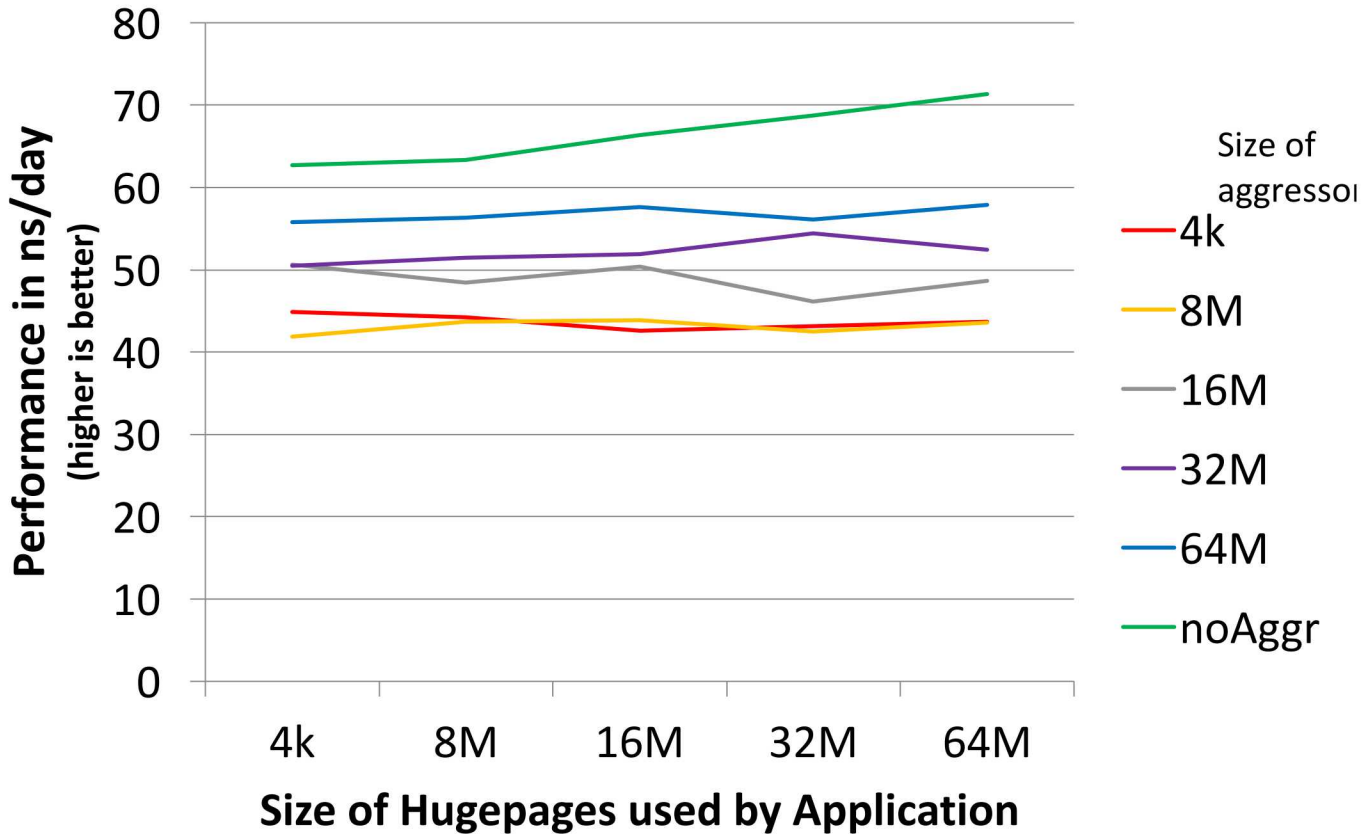


Runtime analysis and visualization of shared network congestion metrics. Backpressure in the network due to multiple applications' traffic limits the injection rate, reducing application performance. Shown: Percentage of NIC's with performance-impacting values.

HLRS Hazel Hen 7,712 node XC40

Goal: identifying “aggressor” and “victim” applications contending for the HSN

Approach: Analyze the system and batch log files. Applications having high runtime variability are classified as “victim” applications and those running concurrently that don’t hit the “victim” variability threshold are considered as possible “aggressor” applications



Performance variance of Gromacs when running in parallel with an ”Aggressor” using different huge page sizes. Gromacs used 2 nodes/blade, the aggressor the other 2.

Outline

- Motivation for Monitoring at **Ten** Large-scale HPC sites
- State of the Practice Examples
- Spectrum of Approaches – common themes and challenges reveal targets for high-value development:
  - Analysis
  - Architecture
- Needs and Requirements for Comprehensive Monitoring
- Conclusions

Analysis and Visualization

- Currently most analyses implemented by sites seek to assess current component state:
  - Fundamental for operations, but not a basic production functionality. Sites have had to develop tests and infrastructure to enable continuous testing, reporting, and response .
  - Sites seek better ability to track and component data at rates that capture events of interest
  - Sites seek increased domain knowledge of expected component behavior and performance interdependencies to better correlate variation with relevant system metrics
  - Sites are willing to offer resources for collaboration with vendors to investigate at-scale and operational issues
- Most log analysis involves detection of well-known log lines and therefore rare and new events go undetected
- Most canned visualization tools enable individual component graphs, which are limited in value and performance for large component counts
- Full system data with a single global timestamp is needed to correlate events across time and space
- Controlled release of data to users is desired, but vendor tools and data access are often privileged

Response

- Notification to users of assessments of system conditions
- Scheduling and allocation based on application and resource state including more fine-grained and dynamic resource allocations and task mappings.
- Useful Response requires increased analysis capabilities and more complex interfaces to schedulers and component/subsystem controls (e.g., downclocking components)

Outline

- Motivation for Monitoring at **Ten** Large-scale HPC sites
- State of the Practice Examples
- Spectrum of Approaches – common themes and challenges reveal targets for high-value development:
  - Analysis
  - Architecture
- Needs and Requirements for Comprehensive Monitoring
- Conclusions

Data Sources

- Need to access and integrate a variety of data sources and types: numeric and text, raw data, derived data, test results, analysis results, off-platform (e.g., facilities)
- Data can come from the system, applications, and external sources

Use of Undocumented and Unsupported Data Paths

- Vendors often develop proprietary solutions to provide telemetry for fundamental system operation but are not meant to be utilized by operations staff as data sources
- Sites have utilized unpublished and/or unsupported codes and APIs.
  - Access data that is either not exposed, or exposed by methods which may be difficult or inefficient to use for the intended analyses.
  - Vendor translation/filtration of data may result in less usable forms of data.
- If vendors would provide in depth documentation of such capabilities and user accessible APIs for reading data that is already being produced:
  - Sites could collaboratively develop and share tools
  - Enable abstractions that would facilitate sharing across multi-vendor platforms

Data Transport For Storage and Analysis

- Absence of a generally accessible transport mechanism and storage solution hampers progress and limits sharing tools
- Different sites have made different choices among the plethora of available data transport and related storage mechanisms
  - Reflect data type, variety, and fidelity and associated performance requirements; directly applicable analysis tools; and supported visualization tools including dashboard displays.
  - Multiple transports may be necessary and even desirable
- Sites must make a substantial up-front effort in design and implementation for fundamental monitoring capabilities and which might not then be a drop-in at other sites

Data Storage and Formats

- Various choices with tradeoffs in insertion rates, query performance, and analytical support
  - MySQL, PMDB (PSQL), ElasticSearch, Splunk, InfluxDB, PSQL with TimeScaleDB etc
- Support for long-term analysis: Storage methodologies which enable keeping near-term data in performant storage, and complete long term data in perhaps less-performant storage which can be reloaded into active data are desired.
- Solutions must address both the mechanics of the archiving and reloading and tracking the locations and contents of archived data
- Estimates of data sizes and ingestion rates have been lacking, based on log size and only vendor-obtained numeric data
- Understanding of the intended queries, analysis, and analysis tools is essential for appropriate design

Outline

- Motivation for Monitoring at **Ten** Large-scale HPC sites
- State of the Practice Examples
- Spectrum of Approaches
  - Analysis
  - Architecture
- Needs and Requirements for Comprehensive Monitoring – inform overall design and product decisions:
  - Architecture
  - Data Sources
  - Data Storage and Formats
  - Analysis and Visualization
  - Response

Architecture

Needs

- We will always need additional data. We will always need to use it in unanticipated ways. We will always need higher fidelity data.
- We will need to direct the data and analysis results to multiple consumers (e.g, users and administrators, various analysis capabilities).
- We will need to integrate the data with other non-platform data.

Requirements

- Vendors should provide well-documented interfaces for accessing raw data at maximum fidelity with the lowest possible overhead.
- Platform owners should be able to determine the data access, transport, storage, and performance tradeoffs of their own choosing, rather than vendors limiting accesses or amounts. The monitoring system, both hardware and software, should be provisioned with this in mind, with options for scaling up. Where access and transport of data might incur impact, that impact should be well-documented.
- Multiple flexible data paths should be anticipated, with changes in data direction and data access easily configured and changed.
- All monitoring system capabilities should be production capabilities and documented, exposed, and supported as such.
- Tools to transport and store the data in native format are highly desirable.
- Extensibility and modularity are fundamental to support evolutionary development.

Data Sources

- We will need production-level insight to the compute platform and all supporting subsystems. This should reveal state of health, utilization, performance, and performance-impacting events.
- We will always need more raw information and we will need information that leverages domain knowledge the platform supplier has about its own architecture and about other vendor supplied components.

- Potential data sources include traditional text (e.g., logs), numeric (e.g., counters) sources, as well as test results and application performance information.
- Vendors should expose all possible data sources for all possible subsystems.
- The meaning of all raw data should be provided. Computations required to extract meaningful quantities from raw data should be defined.
- Continuing interaction throughout the lifetime of the machine with vendor engineering staff is needed for understanding the data in the context of the architecture.

Data Storage and Formats

- We will need to keep all data and to analyze historical data in conjunction with new data in unanticipated ways.

- Easy access to historical data and the ability to access historical data in conjunction with current data is required. As with the storage of application data, all storage does not have to be equally performant; hierarchical storage models with the ability to locate and reload data as needed are desirable.
- Analysis results should be able to be stored with raw data.

Analysis and Visualization

- We don't expect the platform supplier to provide all analytics, nor to know all the analytics that we might need now and in the future. We will need to develop investigatory analyses and visualizations.
- We will want to support both immediate feedback and post-processing analyses.
- We will want to assess application performance in conjunction with system performance and utilization measures.

- Analysis capabilities should be supported at variety of locations within the monitoring infrastructure (e.g., at data sources, as streaming analysis, at the store, at points of exposure to consumers).
- The data store should be designed to support arbitrary extractions and computations. Stores with interfaces that support popular computational packages are desirable.
- Concurrent conditions on disparate components should be able to be identified.
- High dimensional and long term data need to be handled in analyses and visualizations.
- Visualization interfaces and tools should facilitate easy development of live data dashboards.

Response

- We will need to take action on the results of the analysis. This includes feedback to both humans and software.

- Reporting and alerting capabilities should be easily configurable. These should be able to be be triggered based on arbitrary locations in the data and analysis pathways.
- Data and analysis results should be able to be exposed to applications and system software.

Conclusions

High-level generalized experiences from monitoring development on a variety of platforms:

- Component (hardware/software) performance metrics are not well-documented, not published, or both. This includes both access methods and how they can be used/combined to provide insight into resource state and utilization.
- Interfaces for making large-scale data generally available are hard and enabling data exploration is even harder.
- Currently available storage (database) technologies do not lend themselves to the wide variety of use cases for aggregation and analysis of information (combination of event, text, numeric time series) with respect to capacity performance, and size.
- Vendor tools don't generally integrate well with user developed tools across platforms.
- Vendor engineers in many cases don't have answers for system behaviors and performance-impacting issues and large-scale collaborative experiments may be the only path to discovery. Such experiments are costly and must be engineered thoughtfully before execution.

Conclusions (cont’d)

- Site-specific researchers and operations staff work with vendor engineers via back channel relationships to acquire undocumented information which they then use to build one-off tools. These typically cannot be shared for reasons including bilateral NDAs, site reluctance to maintain and/or tools based on fragile or unsupported vendor code with no guarantee of being carried forward.
- Site-developed capabilities may not even be fully utilized at the sites themselves, since on-site personnel may not be able to utilize individual site-developed diagnostics if they don't enable a general, global diagnostic procedure.
- Tools are often developed by/for administrators with root access and ubiquitous ``need to know''. Adding infrastructure to control information access per user is often impractical and hence information that might help users cannot be shared with them.
- Successful monitoring will increase the demand for monitoring. End-to-end monitoring development will be a continuous, evolutionary process, and extensibility and modularity of all monitoring components’ designs will be essential.