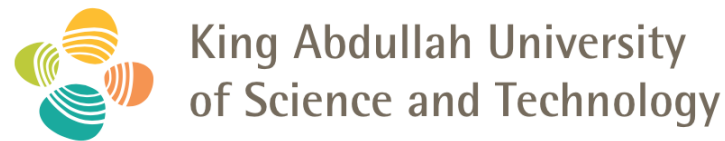


Cray System Monitoring

Successes, Requirements, and Priorities



Motivation for Monitoring

- Different people with different requirements need to monitor the same system
 - Code developer – assess performance impact of algorithmic changes
 - Sys admin – assess and confirm data integrity and security; ensure the system is fully operational
 - System architect – understand overall utilization, performance bottlenecks, and interactions of complex subsystems
- Fear of negatively impacting the system, or spending time developing a “support” service, can limit effort that vendors put into designing monitoring solution
 - This can result in monitoring treated as an add-on, often implemented by the sites themselves
- Demonstrating utility of monitoring overcomes its stigma and encourages vendors and sites to work together to make more effective systems

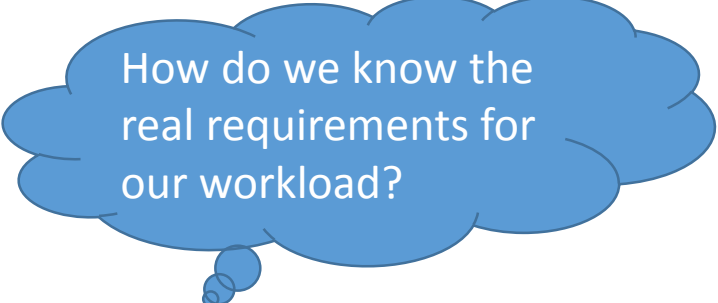
Lots of Subsystems Creating Lots of Data

- SEDC/PMDB
- Standard logs
- Status and health data
- GPU data
- HSN data
- Memory subsystems (MCDRAM)
- Other subsystems such as facility info or parallel file systems

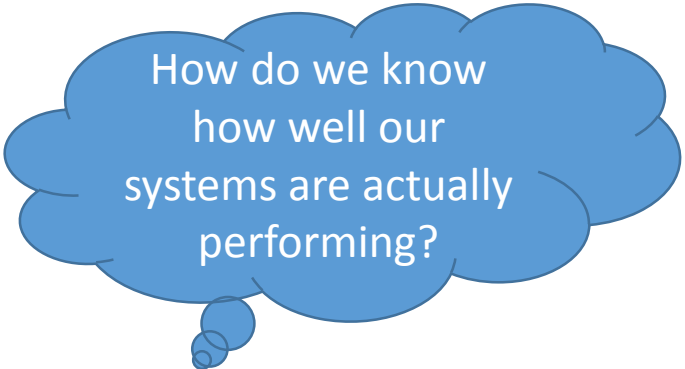
A user of an integrated HPC system must be able to take data from one or more of these sources and correlate them to draw insight and improve their use of the system

Enabling new insights through monitoring

- Monitoring can provide insights into application resource utilization, system state, and contention for shared resources to improve performance, utilization, and throughput
- Focal areas of monitoring investigations of our panel:
 - **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



How do we know the real requirements for our workload?

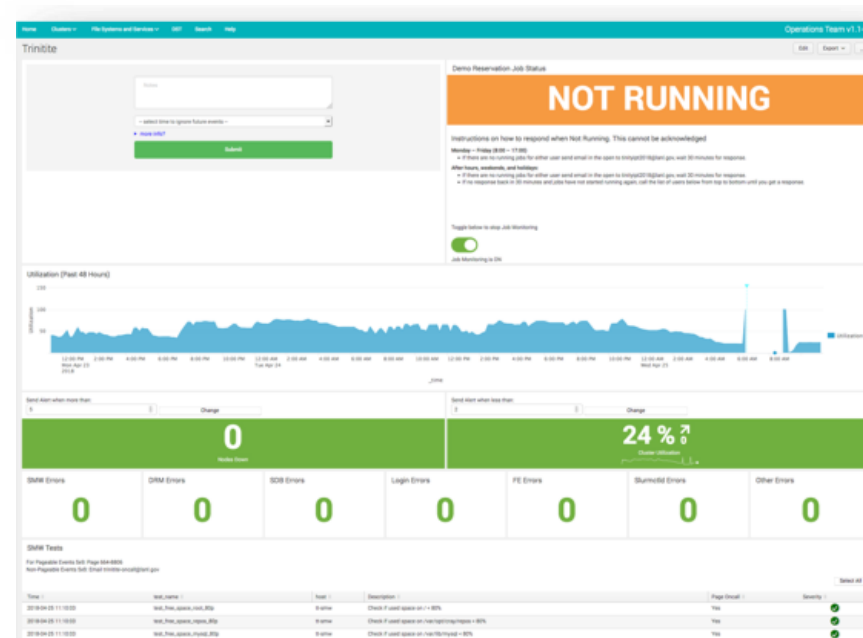
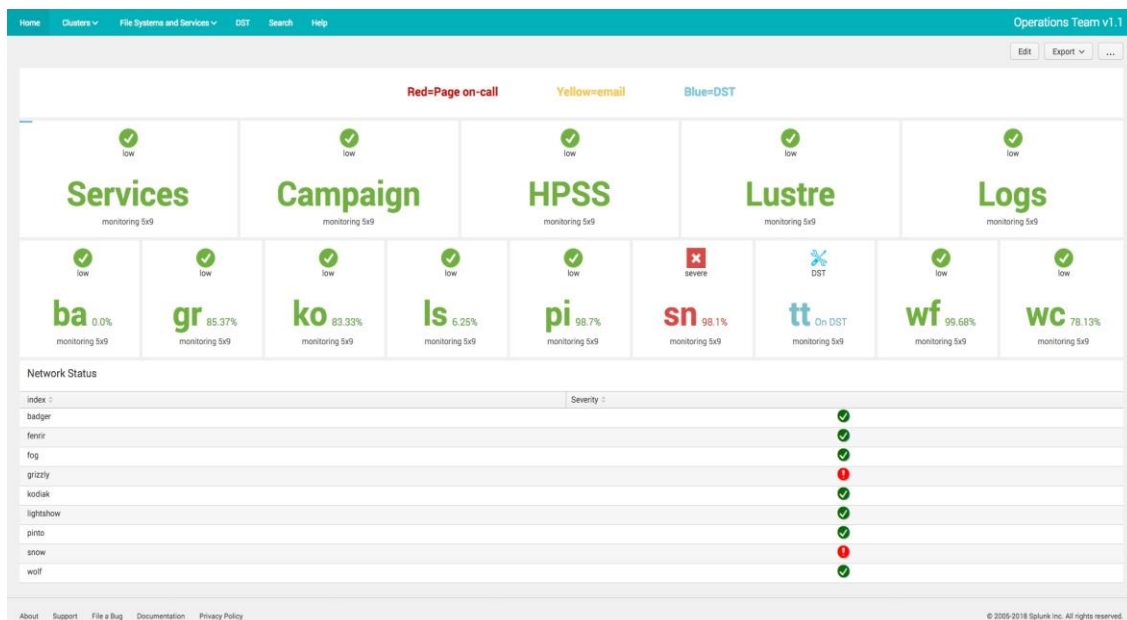


How do we know how well our systems are actually performing?

Ten sites present successes in improved understanding, performance, and operations as a result of monitoring and analysis

LANL: 24/7 Monitoring with Splunk through Continuous Testing

- **Motivation:** Keep Trinity (20K node premier system) up and running jobs 24/7
- **Approach:** 70+ health and performance tests run continuously throughout the cluster. Integrated Splunk dashboard with user interaction lets Operators keep track of everything 24/7.
- **Benefit:** Single pane/Standardized monitoring for entire datacenter



24/7 Monitoring with Splunk through Continuous Testing

- What new capabilities/understanding has your monitoring enabled?
 - Integrated tests suite results into our existing Splunk Operations Application
 - Provide a holistic view of system state
 - Allow 24/7 Operations staff to acknowledge (temporarily suppress) any test failure through Splunk
- What was the most significant roadblock/gap you had/have to overcome?
 - Identifying new failure signatures is an ongoing effort based on new failure modes
- What are your next steps?
 - Add Aries network health tests
 - Add Datawarp health tests
 - Track file system anomalies

CSCS: GPU Diagnostics and Monitoring

- **Problem:** Successful use of GPUs for scientific computing has outpaced the growth in integrated diagnostics, monitoring, and reporting capabilities.
- **Approach:** Collect the current state information from the GPU and error indicators from system log. Analyze the results for errors and symptoms of problems. Integrate this into the Prolog and Epilog processing to provide the capability for automatic actions, including the removal from service.
- **Benefit:** Single error hit on a GPU results in automatic removal from service thereby preventing additional batch jobs from landing on a suspect GPU. Testing in the Prolog verifies a healthy GPU prior to using it for computational work.

GPU Diagnostics and Monitoring

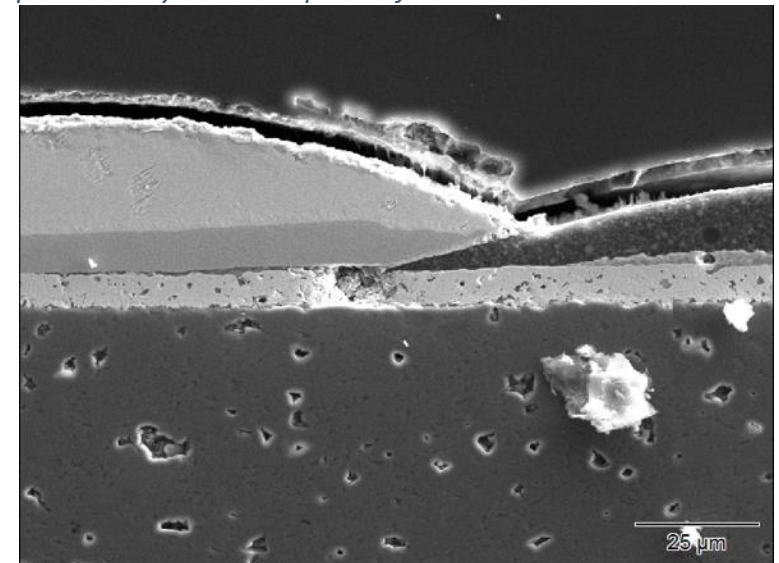
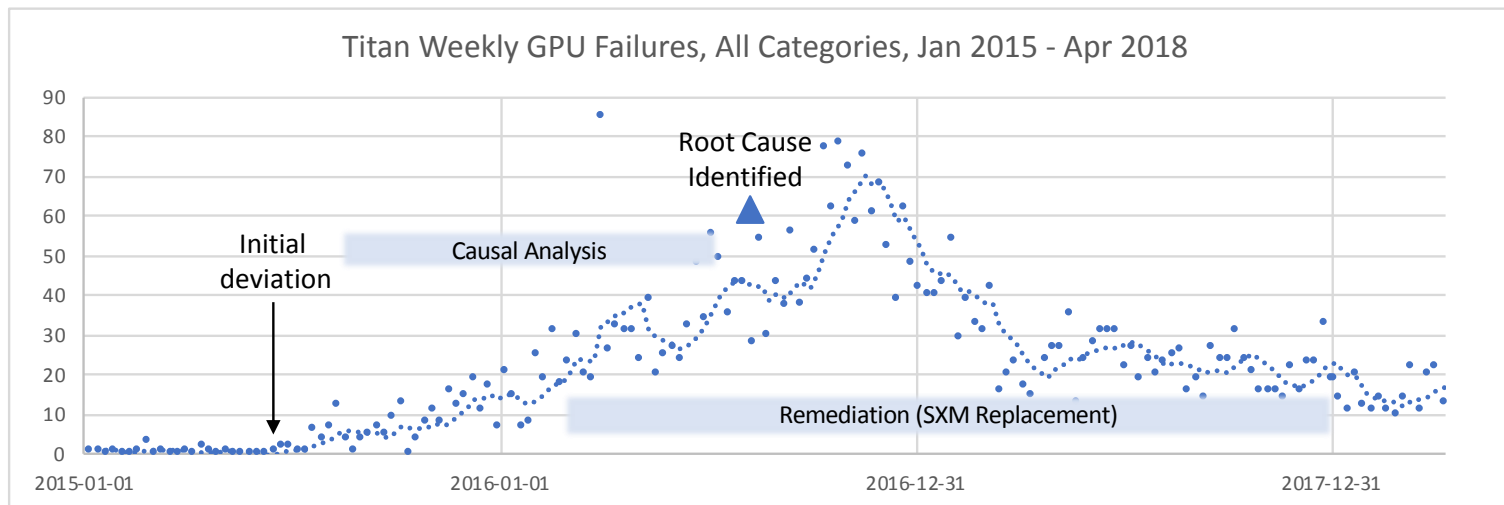
- What new capability/understanding has your monitoring enabled?
 - New failure modes have been identified that are not reported as device errors.
 - “fingerprints” of a problem have been identified.
 - Problems are now experienced by only the batch job that encountered the failure. With automated actions and reporting, the problem is known and often under investigation before a user reports a potential problem.
 - Improved user satisfaction.
- What was the most significant roadblock/gap you had/have to overcome?
 - Identification of failure modes is based on experience, not all is documented
 - Not all problems are reported as errors!
- What are your next steps?
 - Work with vendors to build a supported and integrated solution
 - Investigate utilizing a database for long term tracking and reporting

ORNL: Increasing Long-Term System Reliability

- **Problem:** High rates of DBEs on Titan's GPUs challenge system reliability at midpoint of ORNL's Titan lifetime
- **Approach:**
 - Correlation of multiple log sources exposed abnormal volumes and rate of failure of DBEs
 - Root Cause Analysis identifies supply chain weakness related to the use of non-ASR components
- **Benefit:** Improved reliability of systems that can easily be implemented via procurement requirements and manufacturer process improvements



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



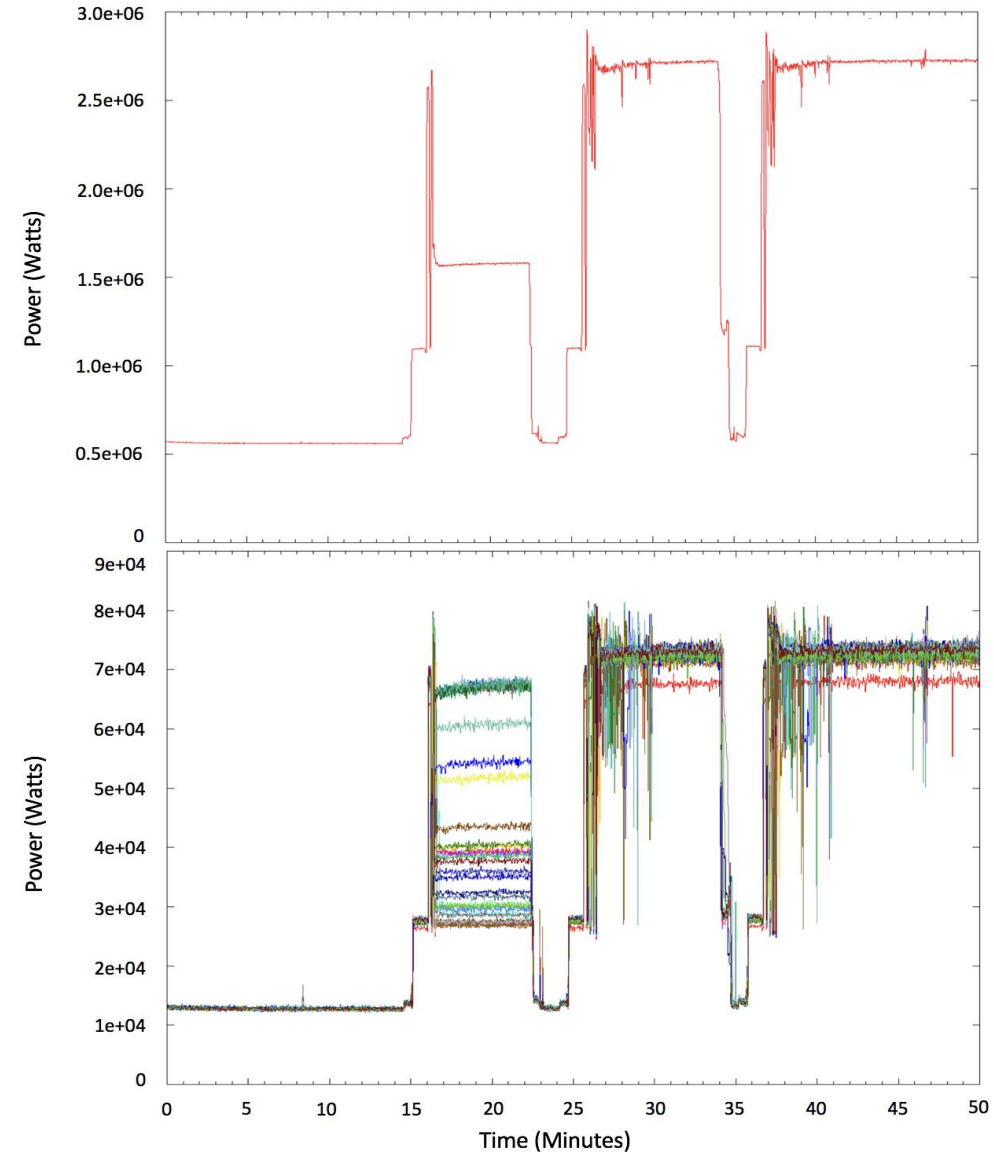
A non-ASR resistor cross-section demonstrating failure due to the formation of silver-sulfide

Increasing Long-Term System Reliability

- What new capability/understanding has your monitoring enabled?
 - Improved awareness/tools for correlating reported failures; symptoms to understanding.
 - Significantly increased system reliability for Titan for the second half of its anticipated lifetime (09/19).
 - Valuable lessons learned for subsequent procurement, easily implemented, that benefit all consumers of these end-products
- What was the most significant roadblock/gap you had/have to overcome?
 - Environmental assessments, temporal and spatial analysis were time-consuming and inconclusive
 - Root cause analysis required special tools and capabilities (cross-sectioning, microscope resolution, x-ray photoelectron spectroscopy, sputter etching...)
 - Substantial financial impact to ORNL and its suppliers to restart the production line for the failing component
- What are your next steps?
 - Use of ASR confirmed for Summit (successor to Titan to enter production in 2019)
 - Consistent use of updated procurement requirements for computer equipment
 - ensures that the Bill of Materials requires use of ASR components,
 - extends liability for all subcontractors for the full lifetime of the system
 - How can we ensure that this capability can be extended for use on subsequent systems, and outside ORNL?

KAUST: Monitoring Power Usage

- **Problem:**
 - Power and cooling constraints in their data center.
- **Approach:**
 - Retrieve data through live monitoring using xtpget and queries to the PMDB during application runs.
 - Build system and application power profiles. Examine for imbalance and irregularity.
- **Benefits:**
 - Make decisions on which applications can operate efficiently under reduced power budgets.
 - Detect variety of issues such as I/O bottlenecks, power imbalance across cabinets.
 - Identified applications that reached higher peaks of power than HPL.

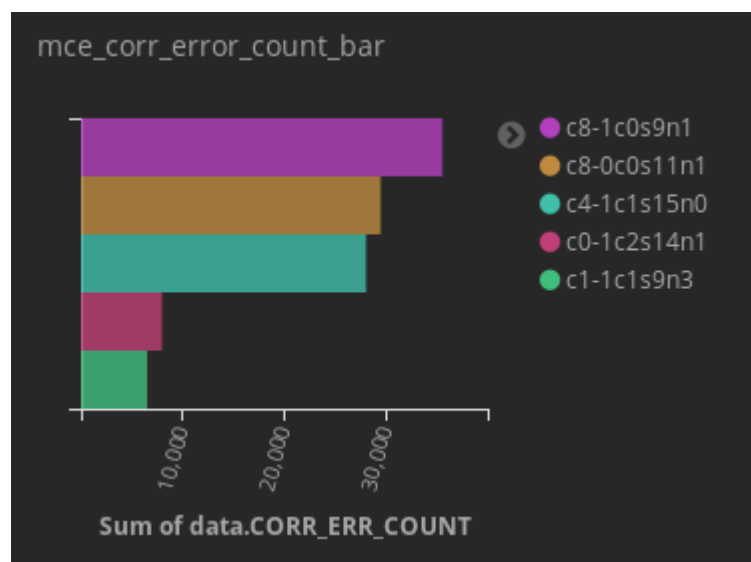


Monitoring Power Usage

- What new capability/understanding has your monitoring enabled?
 - Capability to detect hardware and software(SLURM, CAPMC) issues.
 - Capability to quickly detect performance issue for large scale code.
 - Better understanding of various application performance depending on power capping.
 - Capability of tuning dynamically power capping parameters using SLURM dynamic power scheduling.
- What was the most significant roadblock/gap you had/have to overcome?
 - Only privileged users (sys-admin) have access to the data.
- What are your next steps?
 - Make live monitoring available to users
 - Benchmarking with power profile for future procurement will be mandatory.

ANL: Monitoring via ERD

- **Problem:** hwerrlog files are in binary, making them difficult to ingest
- **Approach:** Read data directly from the ERD, before it reaches a log file
- **Benefit:** Insight into hardware state



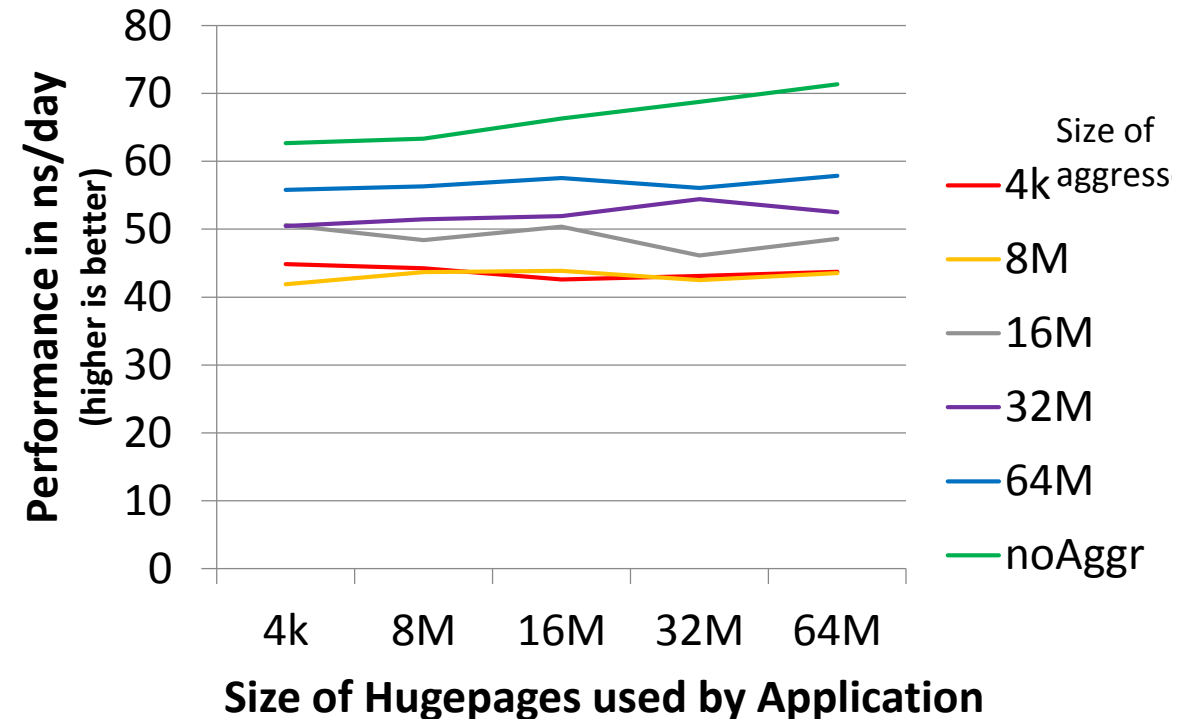
A 24-hour bar graph showing total correctable error counts from compute nodes

Monitoring via ERD

- What new capability/understanding has your monitoring enabled?
 - Visualizations and real-time streaming of hwerrlog data into Elasticsearch and Kibana provide instant feedback on CPU and Aries issues.
- What was the most significant roadblock/gap you had/have to overcome?
 - Lack of API documentation for the ERD
- What are your next steps?
 - Ingest more data via the ERD
 - More sophisticated processing of what we do have

HLSRS: Application Performance Variations

- **Problem:** Performance variation of parallel applications is a critical problem, which prevents users from doing correct performance analysis and planning the machine time usage.
- **Approach:** By analyzing the system and batch log files using Spark we find interactions between running jobs and are so able to solve or reduce the issue
- **Benefit:** Run times getting more consistent between jobs. Also gives Cray insight in potential system improvements



Example : 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.

Application Performance Variations

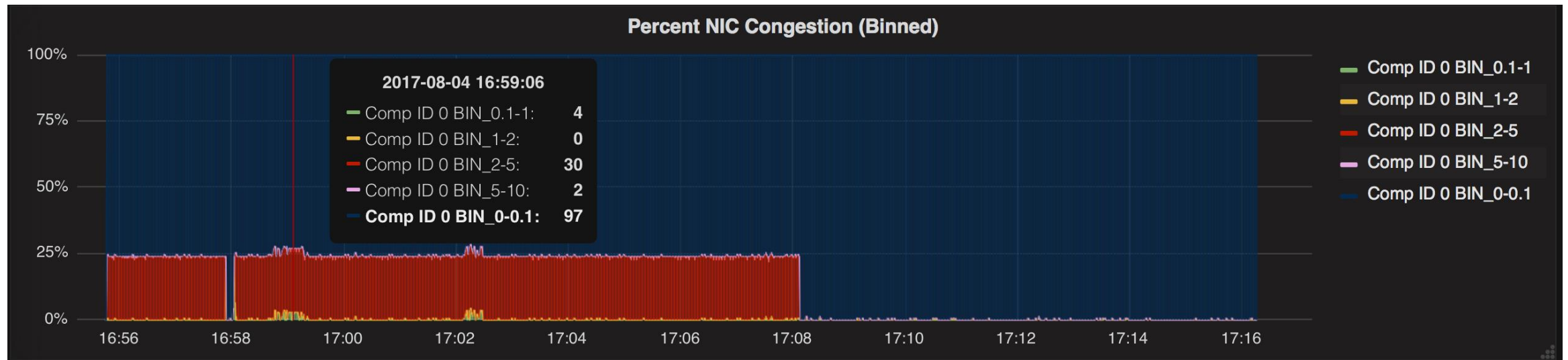
- What new capability/understanding has your monitoring enabled?
 - Found several applications which caused performance variations for other applications.
 - Also found system settings causing problems (LNET patch)
 - In general, 'fixing' the aggressor not only help the 'victims', but also the 'aggressors' itself.

- What was the most significant roadblock/gap you had/have to overcome?
 - Log files only available for administrators
 - Log file inconsistencies, not only from the system, but also from the batch system
 Maybe a common format like JSON could increase the quality of log files

- What are your next steps?
 - Integrate the analysis with Aries Hardware counters using LDMS to collect them.
 This might give us the possibility to move from a post- to live-analysis

SNL: HSN Monitoring

- **Problem:** Performance variation experienced but a) congestion metrics and values associated with impact are unknown and b) conditions in the network are unknown
- **Approach:** Collect whole system network performance counters at intervals $O(\text{sec})$. Runtime computation and dashboard of potential congestion measures (collab. with Cray).
- **Benefit:** Notify users of adverse conditions. Quantify impact. Improve throughput via intelligent scheduling and allocation.



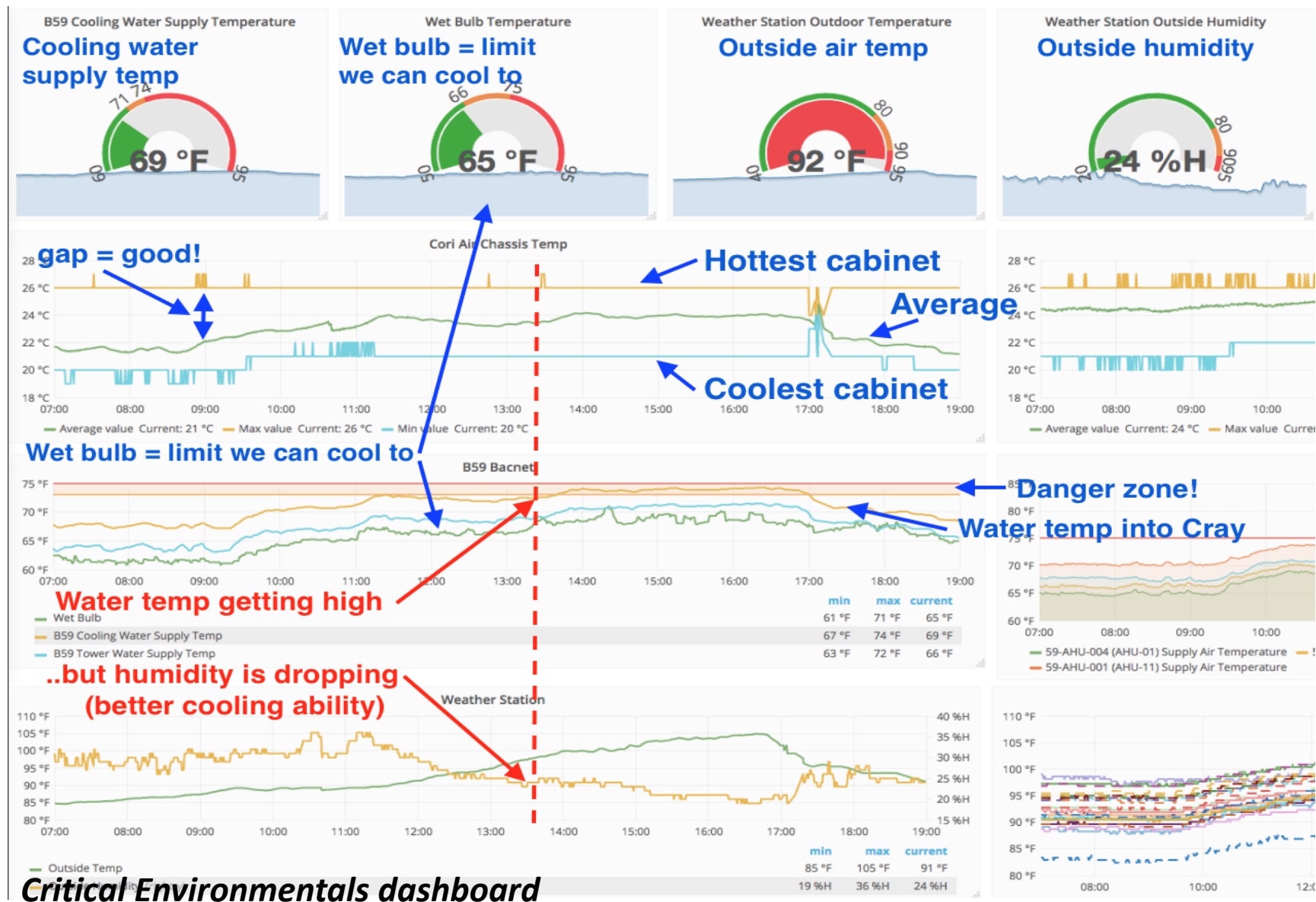
Runtime analysis and visualization of shared network congestion metrics. Backpressure in the network due to multiple applications' traffic limits the injection rate, reducing application performance. Shown: Percentage of NIC's with performance-impacting values.

HSN Monitoring

- What new capability/understanding has your monitoring enabled?
 - *Our visualizations are providing our first insights into the state of the shared network (even without the response part yet).*
- What was the most significant roadblock/gap you had/have to overcome?
 - *Functions of Metrics and their values that can be associated with application impact are largely unknown. (Still are!)*
 - *Attribution of sources of shared-resource contention can be difficult*
- What are your next steps?
 - *Collaboration with Cray and UIUC/NCSA on analyses to characterize and classify state of the network (difficulties: multi-variate, many components)*
 - *Working on analyses to associate that state with application performance in order to identify actionable metrics and values*

NERSC: Environmental monitoring

- **Problem:** Heatwave. Beyond cooling system design point. We cool via outside air, forecast suggest too hot + humid for system to shed sufficient heat
- **Approach:** System environmental + facilities + weather monitoring all on one dashboard
- **Benefit:** If water, cabinet temps stay ok, we can avoid full shutdown. But if we don't know, we have to play safe

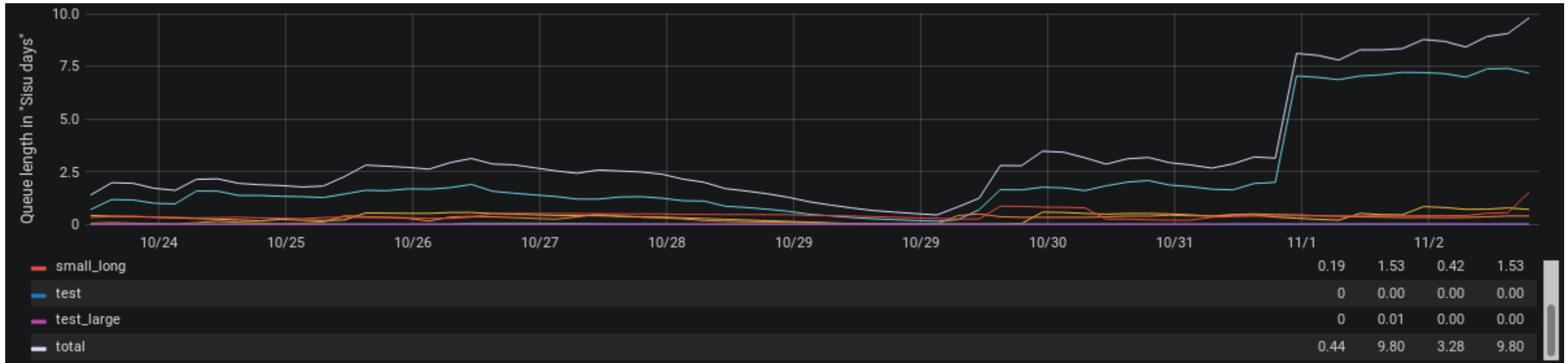


Environmental monitoring

- What new capability/understanding has your monitoring enabled?:
 - Integration of disparate data sources for visual correlation
 - Less-disruptive heat-load reduction was sufficient (drain nodes, kill some jobs).
 - No full shutdown
 - We learned that our system is more robust than we anticipated
- What was the most significant roadblock/gap you had/have to overcome?
 - Requires infrastructure external to the Cray to store and display
 - Datastore / Grafana / Data sources with plugins for ingest (eg we have SEDC, similar for weather, building monitoring)
 - NERSC has large infrastructure for this, but small would be sufficient
 - More discussion in BoF
- What are your next steps?
 - There may be scope for some automation across resources
 - Machine learning to combine sources

CSC: Queue length monitoring

- **Problem:** Inadequate understanding of the amount of work in the queue and of the historical development of the situation
- **Approach:** Collect the total CPU time scheduled for execution and present it in easy to understand units in a flexible graphical user interface
- **Benefit:** Better picture of the queue situation and of its development over time



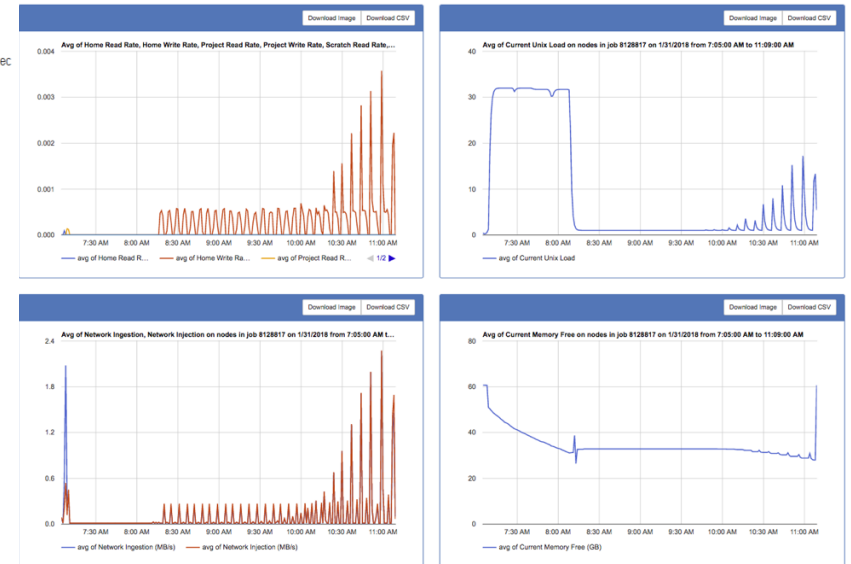
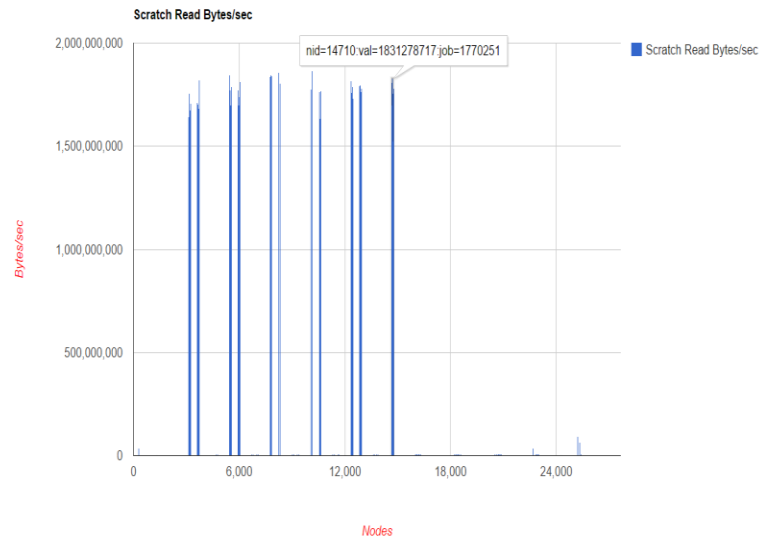
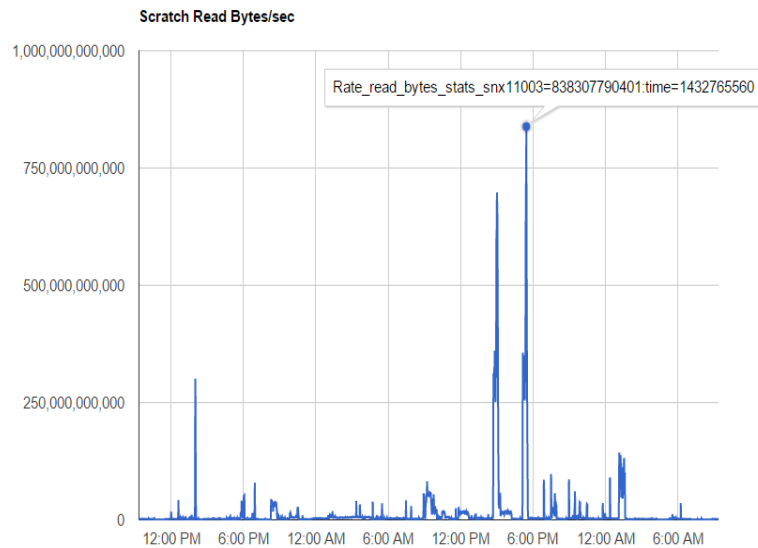
Queue length development presented in a Grafana view

Queue length monitoring

- What new capability/understanding has your monitoring enabled?
 - *With the monitoring in place, we now have a good picture of the queue length with just one glance at the graphs*
- What was the most significant roadblock/gap you had/have to overcome?
 - *Bulk of the work was modifying the Nagios plugin we use to monitor Slurm to include the CPU time and to setup a Graphana view for displaying the data*
- What are your next steps?
 - *We are looking into also graphing the exit statuses of ending jobs to monitor any abnormal bursts of jobs failing or timing out that might hint about system problems*

NCSA: Job Analysis

- Challenge: *Diagnose system behavior anomalies caused by applications.*
- Approach: *View system aggregate metrics over time to find abnormalities. Drill down on times of interest within a metric to show contributing nodes. Overlay job/user information on data to make correlation to suspect workload. Further drill down on workload.*
- Benefit: Fast and low labor mechanism to explore metric data for diagnosing and identifying disruptive workload. Also provides mechanism to show metrics of interest for a job with a suspected problem.



The first graph looks at total system I/O and selects a point in time to reveal the nodes and job responsible for the peak in reads. From there, the suspect job is identified for deeper analysis including more metrics if necessary.

Job Analysis

- Success:
 - **Reassemble** monitoring data with job and user context applied.
 - Quickly **navigate** multiple nodes and multiple metrics, live or historical
 - Quickly **analyze** system anomalies to **identify** disruptive workload.
 - Quickly **analyze** job anomalies to **identify** issues within a job.
 - **Share** information with user.
- What was the most significant roadblock/gap you had/have to overcome?
 - Large amounts of data to store and query- easy to congest backend.
- What are your next steps?
 - Restructure data store and data transport mechanism for capacity, resiliency, performance
 - Restructure front end and backend to use community supported tools if applicable
 - Attempt to find a seamless bridge between “hot” and “cold” spool metric data

Requirements for Monitoring

Design systems where monitoring is a core capability of the system and all its subsystems

Make Data Available

- Expose data in an easy to access and consume format
- Make data available in customizable subsets/frequencies
- Enable exposure of data to multiple users/programs

Sites won't know a priori everything they'll need data to do, and vendors can't guess at all possible use cases.

Enable easy implementation of unanticipated analyses

Make Data Usable

- Provide base set of capabilities
- Data should be well-documented
- Ensure all subsystems in system are time-synchronized

Provide base capabilities so not every site has to roll their own from scratch. Enable sites to understand and use the data in unanticipated ways.

Provide base capabilities & enable sites to expand on them

Conclusion

- Lots of complex sub-systems interact while jobs are running on a system
- Can't just monitor individual subsystems in a vacuum (PMDB over here, Lustre monitoring over there, and never the twain shall meet)
- How can we correlate data from all the subsystems and get actionable insight from it?
- Current lack of standardized exposure of data and common APIs prevents the community from easily sharing solutions
- Cray and the user community need to work together to address this
- **Make monitoring a core capability of the system**