Cray System Monitoring: Successes, Requirements, and Priorities

Ville Ahlgren[†], Stefan Andersson[§], Jim Brandt[x], Nicholas P. Cardo[‡], Sudheer Chunduri[*], Jeremy Enos[**], Parks Fields[‖], Ann Gentile[x], Richard Gerber[††], Joe Greenseid[xi], Annette Greiner[††], Bilel Hadri[¶], Yun (Helen) He[††], Dennis Hoppe[§], Urpo Kaila[†], Kaki Kelly[‖], Mark Klein[‡], Alex Kristiansen[*], Steve Leak[††], Mike Mason[‖], Kevin Pedretti[x], Jean-Guillaume Piccinali[‡], Jason Repik[x], Jim Rogers[‡‡], Susanna Salminen[†], Mike Showerman[**], Cary Whitney[††], and Jim Williams[‖]

[*]*Argonne Leadership Computing Facility (ALCF), Argonne, IL USA 60439*
Email: (sudheer|alexk)@anl.gov

[†]*Center for Scientific Computing (CSC - IT Center for Science), (02101 Espoo | 87100 Kajaani), Finland*
Email: (ville.ahlgren|urpo.kaila|susanna.salminen)@csc.fi

[‡]*Swiss National Supercomputing Centre (CSCS), 6900 Lugano, Switzerland*
Email: (cardo|klein|jgp)@cscs.ch

[§]*High Performance Computing Center Stuttgart (HLRS), 70569 Stuttgart, Germany*
Email: (stefan@cray.com,hoppe@hlrs.de)

[¶]*King Abdullah University of Science and Technology (KAUST), Thuwal 23955, Saudia Arabia*
Email: bilel.hadri@kaust.edu.sa

[‖]*Los Alamos National Laboratory (LANL), Los Alamos, NM USA 87545*
Email: (parks|kak|mmason|jimwilliams)@lanl.gov

[**]*National Center for Supercomputing Applications (NCSA), Urbana, IL USA 61801*
Email: (jenos|mshow)@ncsa.illinois.edu

[††]*National Energy Research Scientific Computing Center (NERSC), Berkeley, CA USA 94720*
Email: (ragerber|amgreiner|sleak|clwhitney)@lbl.gov

[‡‡]*Oak Ridge National Laboratory (ORNL), Oak Ridge, TN USA 37830*
Email: jrogers@ornl.gov

[x]*Sandia National Laboratories (SNL), Albuquerque, NM USA 87123*
Email: (brandt|gentile|ktpedre|jjrepik)@sandia.gov

[xi]*Cray Inc., Columbia, MD, USA 21045*
Email: joeg@cray.com

***Abstract*—Effective HPC system operations and utilization require unprecedented insight into system state, applications' demands for resources, contention for shared resources, and system demands on center power and cooling. Monitoring can provide such insights when the necessary fundamental capabilities for data availability and usability are provided. In this paper, multiple Cray sites seek to motivate monitoring as a core capability in HPC design, through the presentation of success stories illustrating enhanced understanding and improved performance and/or operations as a result of monitoring and analysis. We present the utility, limitations, and gaps of the data necessary to enable the required insights. The capabilities developed to enable the case successes drive our identification and prioritization of monitoring system requirements. Ultimately, we seek to engage all HPC stakeholders to drive community and vendor progress on these priorities.**

I. Introduction

High Performance Compute (HPC) platforms are intended to enable unprecedented scientific insights. They incorporate the latest technological advances in processors, memory, and interconnect at extreme scales to efficiently run applications of increasing complexity. Significant investment is made in the procurement of a platform tailored to support the performance demands of a site's workload.

Performance shortfalls can arise from issues such as components that are faulty or subject to manufacturing variation, applications that are using the platform sub-optimally, or contention among applications competing for the same resources. Monitoring can enable diagnosis and detection of these issues, however, its effectiveness depends on the ability to obtain, integrate, analyze, and curate the necessary data. These underlying capabilities are non-trivial to create, as we may not know, in advance, all information that will be needed, nor how to extract it from the raw data.

In this paper, we present progress by a number of sites in gaining valuable insights via monitoring. Main contributions of this work are:

- Use-case driven description of Cray systems' and supporting subsystems' data. Includes exposure, information of interest, and limitations and gaps
- Case studies from sites demonstrating how monitoring has resulted in increased understanding, improved performance, and/or improved operations

- Use-case driven definition of vendor-neutral priorities for monitoring as core capability in HPC design

We begin in Section II with a discussion of the motivations for monitoring and for community engagement in improving monitoring in current and future systems. In Section III we describe available data for Cray systems, subsystems, and supporting infrastructure needed for building tools and diagnostics. Section IV then details a set of case studies demonstrating improvements through successful monitoring. In Section V, we present a community consensus on priority requirements for monitoring as a core capability in HPC design. We summarize our conclusions in Section VI.

II. Motivation

Monitoring has traditionally been thought of as the province of the system administrator who seeks to use point-in-time information to determine the state of health and load of the system and its components. However, increased insight is required into the state, utilization, and behavior of the platform and its workload by a variety of stakeholders. *Code developers, non-administrative users, system administrators and site support, and system architects (or site leads)* all have different goals for insight. A code developer needs to assess the performance impacts of algorithmic changes. A user needs to know how to best use the system to get optimal performance for his/her application and to be assured of the validity of the output. A system administrator needs to understand how to sustain or improve the user experience and keep the system fully operational as efficiently as possible. The system administrator also needs to assess and confirm data integrity and security, particularly if the system is used to process personal data, where monitoring may be required by regulations [1]. An architect/lead needs to understand the overall utilization, performance bottlenecks, and the details of complex subsystems, to define requirements for the next procurement, the design of a new facility, or to give feedback to vendors' roadmaps to help define next generation technology.

Monitoring can provide such insights, but progress has been limited by fears [2] of performance impact, of problems from running non-vendor supplied software, of overwhelming storage requirements, of inability to extract meaningful insights, or of embarrassing revelations about the architecture or operation. As a result, the monitoring system is treated as an add-on by vendors, rather than an intrinsic part of the system hardware and software with the necessary data exposure, access, and control.

Demonstrating the utility of monitoring will help to overcome these fears and engage all stakeholders and vendors to support the necessary requirements to enable better systems through better monitoring.

III. Data

Understanding system and application state and performance requires integration of data from a variety of sources. From the Cray platform, not all sources are innately exposed or transported. Information from supporting infrastructure, such as storage and facilities, must be obtained as well.

In this section, we highlight the data sources of interest that pertain to the cases presented in this paper. We describe the information they provide, their exposure in Cray systems, and limitations and gaps.

A. SEDC and Power data

System environmental data, such as cabinet temperatures, valve openings, etc, are available via Cray's Systems Environmental Data Collection (SEDC) [3]. This data is well-known to the Cray community. In older Cray Linux Environment (CLE) releases, this data was transported over the Event Router Daemon (ERD) to the System Management Workstation (SMW) where it was written into log files. Since CLE 5.2, this data has been preferentially stored in the Power Management Database (PMDB) [4] on the SMW.

This is sub-optimal for computationally-intensive analyses and analyses requiring access to long term data. To facilitate such analyses, Cray developed the off-ERD endpoint [5], which enables forwarding of this data to locations other than the SMW, where it can be further forwarded or stored in a remote instantiation of the PMDB. Currently, this functionality can only forward *all* SEDC data, as opposed to subsets of cabinets of the system to a single remote location. SEDC data is by default collected at 1 minute intervals from the blade and cabinet controllers.

Power data available on XC systems [4] includes point in time power, accumulated energy, and current power cap limit. This data is exposed on-node at 10 Hz, where the data is exposed via a *sysfs* interface, `/sys/cray/pm_counters`. It is also collected out of band by default at 1 Hz, with a possible maximum of 5Hz, along with cabinet-level power data and job information, and forwarded and potentially stored by the same mechanism as the SEDC data. The PMDB is a PostgreSQL database and hence supports queries such as per-job power usage. Additional tools, such as `xtpget` on the SMW provide current, average, and peak system power usage over defined sample periods. The SEDC and power data are not synchronized in time across the different controllers.

B. Error Logs

Error logs are another well-known data source to the Cray community. Log data is forwarded to the SMW where Cray `rsyslog` configurations filter, reformat, and divide the data into different files. In addition, some data streams, such as `hwerrlog`, are in binary formats, which require Cray libraries to convert to text formats.

The Lightweight Log Manager (LLM) [6] is an rsyslog-based interface that enables forwarding of the data streams off the SMW. Many sites use this ability to forward the data to other consumers.

Limited access to the SMW and the variety of data files and formats makes use of this data difficult. Understanding items in the log files may require specialized architectural or system knowledge and public documentation may be limited. For example, error messages in the network or error codes for DIMMs may be difficult to properly interpret. In addition, multiline log messages also make this data harder to parse by 3rd party log-analysis tools.

C. Status and Health Data

System status and health information is provided through a multitude of commands and logs. The Cray XC Series System Administration Guide [7] contains over 50 pages of suggestions and methods to monitor the system, and while extensive, it is still not comprehensive. The Node Health Check (NHC) can be used to execute tests at the end of jobs and act on the results, including passively logging, marking a node down in a scheduler, dumping the node, or rebooting it. Common tests include checking free memory, stray processes, and shared file systems mounts, but NHC can run any script; for example, counting available hugepages to look at memory fragmentation.

Command line tools such as `xtprocadmin` can be used to show node state information as queried from the service database (SDB). However, this only communicates a node's state (up, down, admindown); it does not include information on why the node is in a non-up state. That information can be found in several possible logs, including console, consumer, or messages logs, as well as logging directly onto the node to check status and error messages.

There is no default mechanism that provides continuous system state and health data in an easily consumable or accessible fashion. Fundamental information desired includes up, down (with reason) and basic functional state of all system components. More complex assessments such as contention for the network or file system are also desired.

D. GPU Data

Data is collected from Graphics Processing Units (GPUs) as well as directly from a node's operating system, because problems affecting a GPU could be on either the system side or the GPU side.

Data from a GPU is obtained by using the `nvidia-smi` [8] command to query the device and retrieve data regarding its current state. This data includes Process name on the GPU (if any), Inforom and VBIOS versions, GPU Graphics Clock, CUDA Driver Version, Power Draw, Retired Pages Pending, PCIe Link Width, and GPU Total Memory. The use of `nvidia-smi` to query the GPUs can also provide useful information from its exit status. There are cases where a GPU is visible from the operating system but is no longer responsive to the operating system. The exit status could indicate that the GPU cannot be found or that the "GPU is lost;" both are serious conditions. From the data collected, the overall health of the device can be assessed.

System messages are obtained by scanning `/dev/kmsg` while the node device status is validated through device files. The GPU reports *Xid* errors back to the system for recording at the system level. These are general GPU errors that could indicate issues with an application, driver problems, and/or hardware-related issues.

The combination of data collected from a GPU and the system provides a complete view of the current state of the GPU as well as previously recorded issues with that GPU.

E. HSN Data

On Cray Gemini and Aries systems, the High Speed Network (HSN) is a globally shared resource. The Cray Gemini network [9] is a 3D torus with largely static routing (except in the case of failures and recoveries). The Cray Aries network [10] is a modified Dragonfly with adaptive routing, which makes the issue of detecting, assessing, and attributing contention more complicated.

There are nearly two thousand Aries network performance counters [11] that provide information about latency, counts of flits across interfaces, stalls, and more. The system uses the counters to make routing decisions and for triggering contention mitigating mechanisms.

Counters are typically accessed by users via tools such as CrayPAT [12] or interfaces such as PAPI [13]. The counters are also available on-node via Cray's `gpcd` interface (e.g., [14]) which until recently was not publicly released or supported. This method has interface functions and data structures to aid in querying the counters and iterating through the return values. From node-level interfaces, only the NIC counters relevant to the NIC associated with that node, and only the Aries counters relevant to the Aries associated with that node, are accessible. Data can also be obtained using `xtmemio` [15] on the SMW to query memory regions of the Gemini and Aries. This method requires elevated access.

Information on network component failures and recoveries, congestion mitigation actions taken by the system, and errors are reported in log files.

Currently, there are no Cray-provided tools that provide the desired insight on congestion and its impact on applications. Creation of such tools by sites is difficult for a variety of reasons. Cray documents (e.g., [11], [16]) provide insufficient information to properly interpret counter values and events of interest to easily assess application performance impact. In addition, some essential data is not exposed. There is no information on the source of network traffic. Per-flit routing information is not exposed, therefore

there is no direct way to determine the cause of congestion or if an application will be affected, although some system assessment of these (without the underlying data values) may be provided in the logs in extreme cases.

Due to the global nature of the shared network, querying data via the node interface from within the allocation of an application provides only a limited view of the system's network state that can affect the application's communication routes.

F. MCDRAM Data

Intel's MCDRAM memory interface controller counters [17] are accessible using an on-node performance monitoring interface. They provide information such as the MCDRAM read count, write count, MCDRAM hits and misses, and DDR reads and writes. The register code determines which data is being accessed. `perf_event_open` [18] is a generic Linux interface which can be used to read the *untile* counter registers.

G. Supporting Subsystems Data

Supporting subsystems and external systems also affect the machine. Data from the facilities which feed the cooling infrastructure of the machine is of particular interest. Shared file system appliances, such as ClusterStor, can report a variety of information about the state of the file systems, from utilization to performance information. Power information can be retrieved from smart PDUs in the service racks.

This data is not intrinsically integrated with the machine data. There are complexities in achieving this since the facilities data infrastructure may not be designed for easy export of live data, it may require a special protocol for interacting with the data, and facilities may be in a different security domain than the platform. Tools such as "View" from ClusterStor are not designed to easily stream data to other sources. Other file systems, such as direct attached Lustre, GPFS, or Panasas must be monitored using different tools or mechanisms. Besides data from the client and server processes and servers, there is often an additional storage network fabric, such as InfiniBand or high speed Ethernet, that must be monitored for status and health. PDUs, BMCs, and other sources all have specific commands to interact with them. The ability to monitor all of this data and correlate it to the system data that Cray provides for the XC infrastructure would be extremely beneficial, but at this time, it is left to each site to find a way to access and aggregate the data applicable to the task being addressed. Note that data in this context is meant to imply a combination of numeric and text information (e.g., counters and log messages).

IV. Site Success Stories

In this section we illustrate motivations for monitoring stakeholders, processes, and outcomes for monitoring through the presentation of cases where sites have increased understanding, improved performance, and/or improved operations through the use of monitoring.

A. Continuous Testing

Many sources of degraded job performance are difficult to detect from a node or job perspective at application runtime because the expected performance of any particular application run is unknown. It is unknown because, depending on the application, performance and the underlying resource utilization can vary significantly depending on input parameters. To solve this problem, a number of sites have developed a variety of tests that are run periodically where the expected outcomes have been well characterized. Additionally, some sites have incorporated tools that identify trends and/or problems through continuous evaluation of information from a variety of other sources such as syslog, environmental monitoring, facilities monitoring, the local weather service, etc. With continuous analyses of available information, operations staff and/or users can be quickly alerted, the cause of problems diagnosed, and corrective action taken where appropriate.

1) LANL: In order to ensure Trinity, a 20,000 node Cray XC40 system, is running jobs efficiently and functioning properly, Los Alamos National Laboratory (LANL) has integrated the output of a custom test suite with their Splunk Operations Application. Their current test suite includes over 50 tests, run from privileged components, such as the SMW and DRM node(s), to check and assess the health of a variety of Trinity's many components. These include checks on configurations (e.g., data warp nodes), verification that essential daemons and services (e.g., erd, mysql, firewalls, slurmctld, llmrd) are running, checks that filesystems are mounted, and checks on freespace, among others. Tests are run at 10 minute intervals, via cron, and can also be run on-demand.

This purpose-built Splunk Application combs appropriate test output data and log files for indications of alert-worthy errors or performance degradation. The integration of test results and log information, when coupled with an easy to understand dashboard, has enabled Operations staff to track, and quickly assess, the health of the major datacenter components (Fig. 1). Drill-down dashboards enable at-a-glance assessment of the major components of Trinity (Fig. 2). This 24/7 monitoring system enables Operations staff to take appropriate corrective action or escalate to the system administrators when serious problems are identified.

LANL plans to continue adding to the test suite and has recently added a job monitoring panel to enable tracking of specific high profile user jobs.

2) NCSA: There are significant challenges to understanding how to, in a proactive fashion, identify abnormal behavior, including root causes, in large parallel filesystems. The Blue Waters staff at the National Center for Supercomputing Applications (NCSA) is trying to better understand the

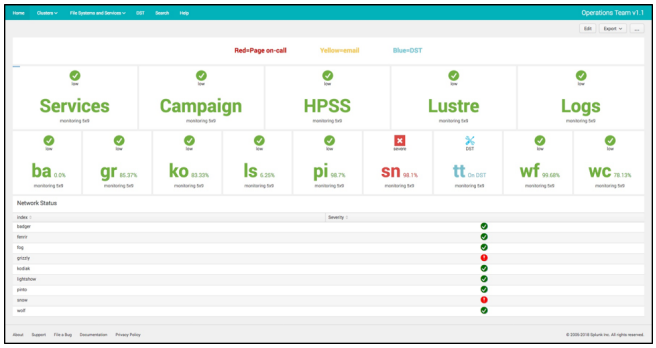

Figure 1. LANL Splunk Operations Grid showing high-level overall state of everything in the datacenter.

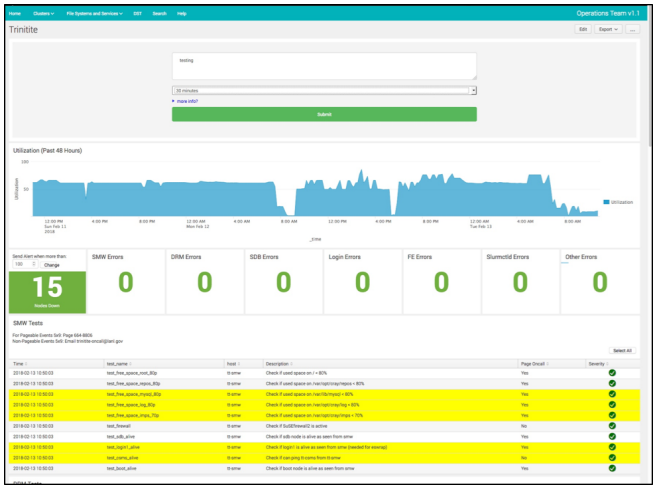

Figure 2. LANL Drill-down example showing result of tests, as well as selected tests for acknowledgment, on Trinity

end user interactive experience with respect to filesystem responsiveness. The common detection of filesystem unresponsiveness in the past has often been through user support tickets. NCSA has been developing an approach to detect filesystem problems before users report the impact. Their solution probes a set of file I/O operation response times, targeting all independent filesystem components from clients representative of the various physical paths to the data and various software environments used for access. This includes measurements to each Lustre OST from within the HSN through Lnet routers, from Import/Export nodes attached via InfiniBand, and Login nodes that can experience impact from competing users.

The NCSA solution uses the periodic creation of small (4k) files specifically located on each filesystem storage target, and the scheduled regular execution of a process that measures the latency to perform the creation (a metadata server test), writing, rmdir, or removal from various locations in the system. This data is gathered on one minute intervals either for every node in a given class (such as login nodes where they want to understand the behaviors of each), or from one node of a class that is sampled from a pool of

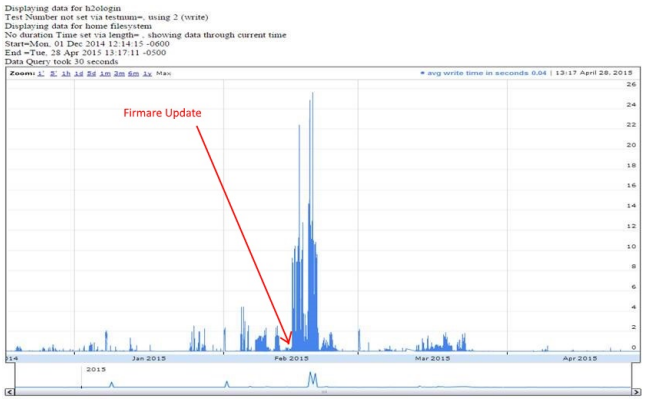

Figure 3. NCSA visualization of average response time for filesystem write response tests. Performance degradation is readily apparent and enables association with the timing of a firmware change. Note that the Y-axis is response time in ms.

nodes in that class. This data is stored in a MySQL database and made available as raw data or graphed for visual analysis with a Google Graphs interface.

Using this methodology, NCSA staff have been able, without adverse impact on running applications, to: 1) diagnose the precise time that configuration changes have had a measurable impact on filesystem response times, 2) recognize filesystem performance problems due to hardware errors that hadn't yet tripped an alert threshold, 3) recognize filesystem usage problems before user complaints have been generated, and 4) measure the impact of backup and filescan tasks that enabled improvements through policy modifications.

The graphical interface was developed to view the data using Google Graphs when they faced a challenge "root causing" a filesystem issue. They began receiving multiple reports of session hangs on the interactive login nodes. It was unclear exactly when the behavior began and how frequently it was occurring. With a visual interface to the data, it was apparent where the behavior changed dramatically. NCSA plotted (Fig. 3) a time series chart of the average response time to perform a 4k write to each filesystem component for each minute. The expected value for normal operation is small number of milliseconds, with infrequent and brief disturbances from large parallel I/O. They were able to pinpoint down to the minute when the behavior changed which aligned with a firmware change to the storage servers. Reverting the software immediately resolved the issue.

Next steps include rewriting the collection service to better address some timeout situations that had been observed as well as improve the code structure to enable generalized deployment at other sites.

3) NERSC: The National Energy Research Scientific Computing Center (NERSC) identifies abnormal or unsatisfactory performance on their Cray platforms with the aid of a suite of compute, communication, and I/O benchmarks that run in an automated fashion and whose performance is tracked over time on a series of plots published on the

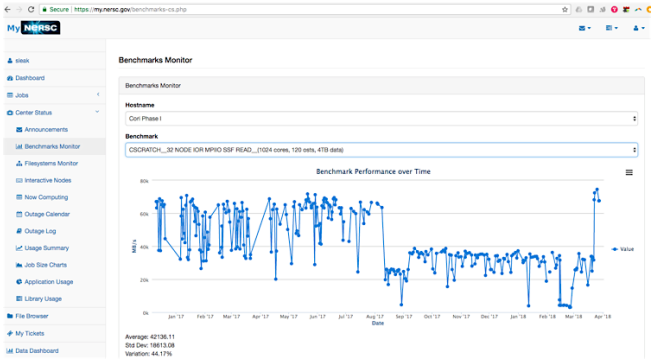

Figure 4. NERSC Benchmarks Monitor dashboard showing performance of a benchmark on the CSCRATCH file system. Occurrence of performance degradation is clearly seen.

https://my.nersc.gov web page.

These tests are selected to exercise a variety of features and subsystems and provide a measure of the health of each subsystem. The tests are indicative rather than full diagnostic tools and sometimes followup diagnostics are needed, for instance by running an application under a performance analysis tool such as CrayPAT.

This methodology helped recently when an applied Lustre patch resulted in a significant performance reduction for an I/O benchmark (see Fig. 4). In this case not only did the benchmark performance tracking provide an early indication of a problem and when it began, but it also enabled narrowing down the cause to I/O on a scratch file system. This, in combination with the time information, enabled direct association of the performance reduction with a system software change. After applying a kernel patch from Cray, the fix was again verified through the results of the tests. Without this continuous testing in place, the problem might well have persisted with the users experiencing a performance reduction with no obvious explanation.

B. GPU

The continued success of accelerated computing on GPUs has driven the need for advanced rapid detection of problems with GPUs. Over the years, the successful computational use of GPUs has outpaced the ability the quickly detect problems and react to them. Both routine problems with the health of individual GPUs, as well as subtle manufacturing process problems, can be identified with appropriate data from monitoring of the GPUs.

1) CSCS: Based on many years of experience with computing on GPUs, the Swiss National Supercomputer Centre (CSCS) has gained significant insight into the many GPU and GPU-related failure modes, and has developed a comprehensive and very efficient suite of tests [19] for the validation of a GPU's health. Error conditions for both the host system and the GPU are examined in order to obtain an overall assessment of the health and status of the device.

Their motivation is based on the philosophy that no batch job should start on a node with a problem, and a problem should only be encountered by at most one batch job – the job that was running when the problem first occurred. Furthermore, the entire process must function in an automated manner. The primary objective is to improve the overall user experience on the system by providing a reliable computing environment.

The general principles for the design were:

1) All tests must complete quickly
2) The test suite must be easily extended for additional checks
3) All checks could be run interactively as a diagnostic verification
4) All errors are recorded
5) Nodes with identified problems are automatically removed from service

An efficient and flexible test harness was put in place that supported an automated batch operation mode as well as an interactive operation mode. The batch mode could be embedded within the system while the interactive mode provided an environment to verify an error and easily recheck for problem resolution.

There are three components of the implementation:

1) Problem Detection
2) Problem Reporting and Recording
3) Automated Action Response to Problems

Checks have been developed to detect issues that could affect performance, as well as hardware-related errors. In some cases, a GPU will continue to function without error or warning, yet performance has degraded.

Tests are executed at both the start and end of each job. Testing at job startup will prevent a batch job from starting on an unhealthy node, whereas testing at the end of the job will prevent further jobs from utilizing an unhealthy node, as well allow for the identification of the batch job that encountered the problem. This is an important point, as errors could be reported that apply to the application that was using the GPU and not necessarily a problem with the device. It also provides advance notice that a user's batch job may have encountered a problem.

Nodes are monitored for any reported Xid error on the system side. There is one major limitation to use of Xid errors captured in this manner - if a node records any Xid errors, the node must be rebooted to clear the error message from the nodes messages. Otherwise, it will continue to be reported after the error condition has been corrected. This has been viewed as a minor inconvenience, as the majority of the recorded errors either represent a true hardware failure or a retired page pending. The corrective action for a pending retired page is to reboot the node.

When a problem is detected, sufficient information is recorded to allow for quick diagnosis of the problem and identification of potential recovery actions. The information recorded includes: Batch Job Identifier, Nodename,

Username, Failing Test Name, Appropriate Error Text, Recommended Recovery Action, and GPU Serial Number. All recorded messages are automatically timestamped. This provides the complete picture of error condition:

- who - the affected user
- what - the GPU serial number
- where - the node name
- why - the error text
- when - the timestamp
- how - the failing test name

GPU serial number is recorded because normal problem isolation techniques include the relocation of suspect devices in order to isolate problems. By tracking the serial number, suspect hardware can be monitored regardless of location in the system or how many times it is shuffled. Another benefit to doing this is for longer tracking purposes. It is possible that a returned GPU could be repaired and returned to the site, and a full history of the device needs to be maintained.

As an example, one test monitors the PCIe link width of the GPU. This is a failure mode where the device has not failed or reported errors, but the device is no longer capable of the delivering the expected performance. The PCIe link status is obtained with the command: `nvidia-smi --query-gpu=pcie.link.width.current`. For this test, the expected response is 16 and any other value triggers a test failure. A reboot of the node will confirm if the problem is persistent and requires the replacement of the GPU. This test detects a clear performance problem that can be identified without a user reporting slow performance.

All of this is tied together in an automated solution. The batch systems prologue and epilogue execute the test suite at each job startup and completion, respectively. If a test failure has been detected, the suspect node is automatically added to a system reservation for suspect nodes in order to prevent further use prior to validation. Additionally, notification of this action is recorded in the system logs as well as a Slack channel.

The end result is a near 100% detection rate of GPU related issues on first hit that triggers an automatic removal of the suspect hardware from production operation. During 2017, there were only four issues that were not detected, which led to the development of additional checks. Using this methodology, the reliability of the system and user experience has been greatly improved.

As new conditions are identified, new tests are incorporated. Performance enhancements of the tests, as well as the test harness, are routinely validated, as this process must not adversely impact the batch jobs or waste excess compute cycles. The next step in the evolution of this process will be to record such failures in a database along with all the details. This will enable easy reporting and trend analysis. Periodically running tests to perform small stress tests on the GPUs are also being investigated. The results would be used to validate that the GPU's computational capability is at the optimal performance level.

NVIDIA has been working on the Data Center GPU Manager (dcgm) which may reduce the number of custom checks required. As the systems software stack moves forward, more analysis of the tool and how best to incorporate it will be possible.

2) ORNL: Titan, the Cray XK7 at Oak Ridge National Laboratory (ORNL), was introduced in Nov 2012 as the fastest, most capable supercomputer in the world. In Jul 2015, just 2.5 years into the expected 5-year minimum life of the machine, OLCF system administrators first identified an alarming trend. Until this time, the rate at which the NVIDIA Kepler GPU SXM modules were failing had a mean time to failure (MTTF) of about 1 per day (MTTF==24 hours). However, GPUs were starting to fail at a slightly higher rate. From Aug 2015 to Jul 2016, the failure rate continued to climb, reaching 3 failures per day (MTTF==8 hours) by Feb 2016 and 6 failures per day (MTTF == 4 hours) by Jul 2016. Each failure causes the application running on that GPU to crash, and calculations had to be restarted from that application's last checkpoint. With very high system utilization ($\sim 92\%$), these failures began to have a significant negative impact on the user community.

ORNL worked with both Cray and NVIDIA to qualify a significant number of potential dependencies, including NVIDIA GPU firmware and software changes, Cray OS and PE changes, environmental conditions in the data center (air flow, supply air temperature, air quality, relative humidity), and temporal and spatial analysis of the individual failures. When did individual errors occur? Where did they occur in the system? What workload was executing on those GPUs at the time of failure? Could any correlation to the position of the GPU within the system, within a compute rack, or on a specific node board be made? Could projections be made relative to which nodes had a higher probability of failure, i.e expected to have a shorter TTF? Could any correlation be made relative to the specific SKU for the part? Could the manufacturing, shipping, delivery, or installation date be correlated to specific failures? What other GPU SXMs were produced at the same time, under the same conditions, with the same SKU, or the same testing? Where were those products distributed globally? Were they failing in other locations at similar rates?

At the same time that the causal analysis was under way, there was a significant amount of effort devoted to the failure analysis for the GPU SXM module. The form factor for that module is a small daughtercard with (primarily) the GPU, GDDR5 memory, and the PCIe interface to the Cray motherboard. NVIDIA examined one of the failed modules, and identified a small number of surface-mount resistors on the printed circuit board (PCB) that did not meet their design-point for resistance, i.e. the measured resistance of the failed part was substantially higher than

the design specification for the resistor. Examination of additional failed modules confirmed similar behavior. The presence of these out-of-specification resistors on the GPU module was conclusively determined to be the key indicator of this class of failure.

Through a complete non-destructive and destructive analysis of both failed and good modules, ORNL, Cray, and NVIDIA were able to identify that NVIDIA's manufacturing process had not used anti-sulfur resistant (ASR) materials on components that required them. This caused a build-up of sulfur-based compounds on modules, as can be seen in Fig. 5), which shows a microscopic examination that revealed the presence of crystalline structures on the (non) anti-sulfur resistant (ASR) surface-mount resistors.

While root cause analysis continued, failures by early Fall 2016 on Titan had approached 11 per day, with an MTTF nearing 2 hours. NVIDIA used all existing failure data, both temporal and spatial, to construct a model to predict and rank the probability of failure for all GPU modules in the system. Over the course of the next six months, 9,500 of the highest ranked (probability of failure) modules were systematically replaced with new stock that used ASR resistors (requiring the restart of the manufacturing process in China). Of the 18,688 modules in the system, more than 11,000 were eventually replaced. The positive impact to the MTTF is substantial, with the failure rate as of Apr 2018 near 2 per day. The timeline of failures relative to the diagnostic and remediative events is shown in Fig. 6.

ORNL continues to monitor the data center environmental conditions, specifically for the presence of corrosive gases such as those that caused the build-up that caused the defective modules to fail.

Equipment manufacturers must use high quality materials in their manufacturing process. The amount of productivity lost by the Titan community while the problem with the GPU modules was being identified was very high. Manufacturers and suppliers should integrate many failure tests, including those that test for reaction with corrosive gases and other physical environmental tests, into their quality assurance process. Accelerated failure tests provide critical reliability data that can be used to more accurately forecast the MTTF of components in the system, making it easier to more quickly identify when there is a problem.

C. Power

The power draw of modern HPC systems can be in the megawatts, and variations can have significant impact on sites' budgets and power providers provisioning decisions. Because of this, sites with large scale resources are beginning to explore ways of increasing energy efficiency of both the system and the applications' use of system resources. In order to accomplish this goal, some reasonable fidelity monitoring of the power profile of a system in the presence of workloads must be performed. In this section we present how two Cray sites are utilizing the PMDB and other in- and out-of-band power monitoring capabilities of the Cray XC40 platform to gain understanding that can enable them to control power draw and minimize waste.

1) KAUST: Due to power and cooling constraints in their data center, King Abdullah University of Science and Technology (KAUST) needed to instantiate power aware scheduling on their Cray XC40 system, Shaheen2, managed by KAUST Supercomputing Lab (KSL). This would enable them to maintain cooling within their power budget. They instantiated two queues, one with 1805 nodes uncapped and one with 4367 nodes capped at 270W. Then, in order to maximize the utilization of the power budget, reduce the waiting time in the queue, allow large scale runs, and dynamically update power capping, they used the dynamic scheduling features of SLURM and Cray's Advanced Platform Monitoring and Control Utility (capmc) [20] to execute the capping. capmc is used with the workload managers and scheduler to set the power policy execution, like the power caps, and to monitor the XC40 power usage, by querying the system power data.

While the main driver for power monitoring was to ensure that the data center did not exceed its allocated power budget, it also proved useful for detecting problems. Problem detection is accomplished by characterizing power profiles of known applications and then comparing their power profiles whenever they are run against the baseline profiles of good runs of the applications. KSL builds power profiles using information obtained through live monitoring using `xtpget` (Section III-A) and queries to the PMDB during application runs. The information utilized includes total power draw, per-cabinet power draw, water valve position, temperatures, among others.

Anomalous power-use behaviors can result from problems either in the system or in application code. In the cases of hung nodes or load imbalance, which indicate problems that will result in resources and power being utilized to no purpose, a job is proactively terminated, saving both compute resources and power. This is shown in Fig. 7, where power usage variation of up to 3 times was observed between different cabinets (bottom) and full system power draw was almost 1.9 times lower during this period of variable cabinet usage (top).

Monitoring also enables comparison of performance of an application with and without power capping. This information can be used to make decisions on which applications can operate efficiently under reduced power budgets.

Characterizing application performance, in terms of power characterization, in conjunction with continuous monitoring of power utilization, has enabled the KAUST data center to run at an increased power budget without risk of violating the limits. It has also enabled detection of a variety of issues such as I/O bottlenecks, power imbalance across cabinets, and excessive delays in srun spawning tasks across

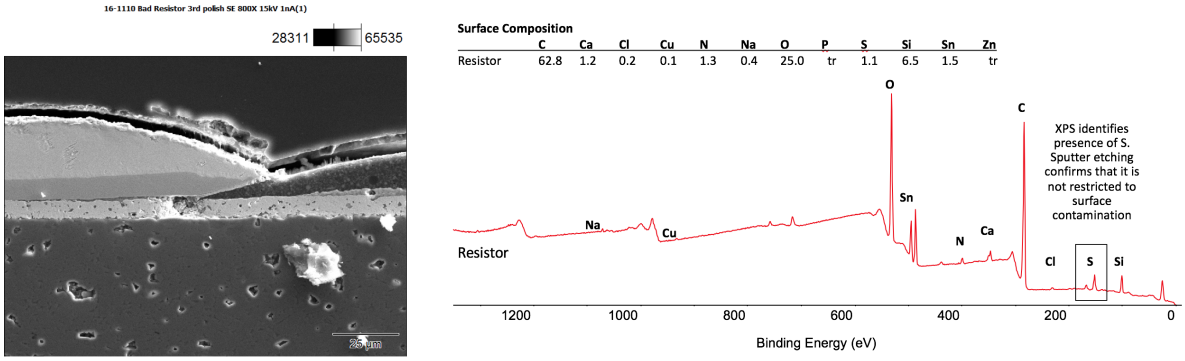

Figure 5. ORNL: Cross-section of failed resistor showing formation of Ag2S as gaps in the resistive element (left) X-ray Photoelectron Spectroscopy positively identifies the presence of sulfur (right).

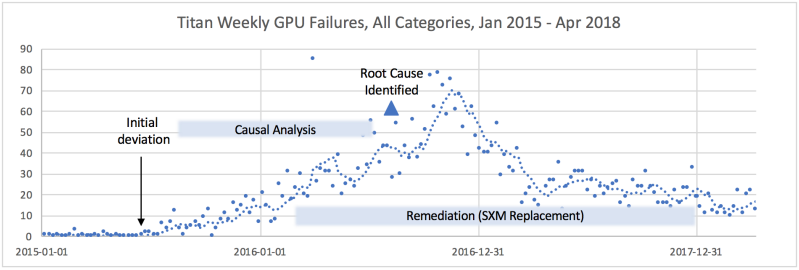

Figure 6. GPU-related failures in time, all categories. Significant diagnostic and remediative events are marked.

the system, when instantaneous power usage is lower than expected.

This helps the KSL staff to detect critical issues with the power setting with SLURM and capmc during system software upgrades, identify applications that are more power hungry than HPL, such as memtest, NekBox [21] (optimized version of Nekbone), and MOAO applications [22].

The main limitations thus far are that users cannot access the monitoring data and that due to the large volume of data captured, it can only be stored for a couple of weeks, making comparisons and trend analysis across long time intervals impossible. It would be desirable if Cray would provide a capability for users to get live access to power monitoring information for their jobs, as HPC center staff cannot provide this manually in a scalable manner, and only getting it after a job terminates removes much of the utility.

2) SNL: Sandia National Laboratories (SNL) research staff are looking for ways to improve the energy efficiency of their workloads. This will ultimately enable them to field larger systems (i.e., more efficient use of limited facility power). Software control of power knobs (e.g., p-state and power caps) is one well-studied way to accomplish this. SNL is leveraging the Cray out-of-band power monitoring infrastructure, in-band p-state control, and out-of-band power capping capability to analyze several mini-applications and the Sandia SPARC application for their suitability to system software-directed power management.

The Cray out-of-band power monitoring infrastructure was configured to measure each node's power draw at 5

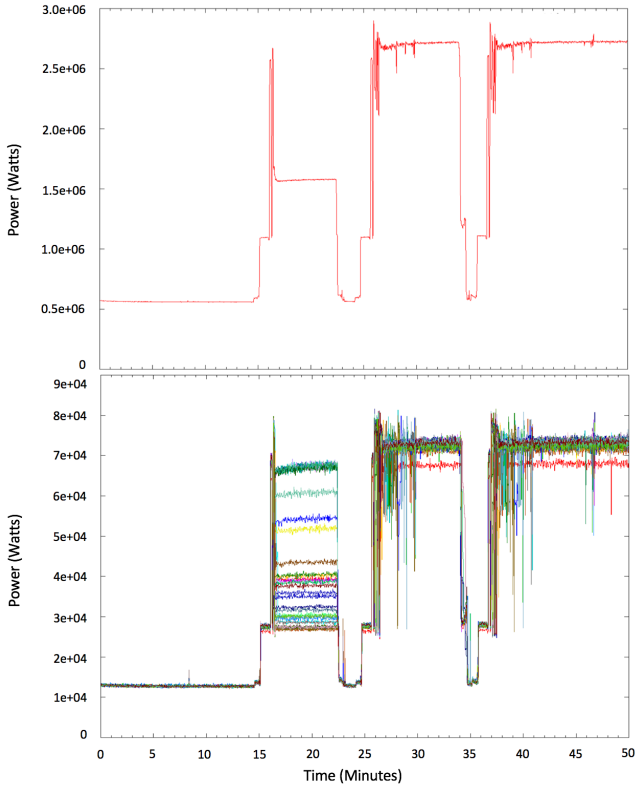

Figure 7. Shaheen overall power usage (top); power usage per cabinet (bottom). Load imbalance issue was detected with power usage variation.

Hz. Each workload was run, noting the jobid and apid. Then post-run, this information was used to query the Cray power management database and generate point-in-time power plots of the run (i.e., time on x-axis, power on y-axis, one curve per node). By sweeping configuration parameters such as p-state, power cap, node type (Haswell vs. KNL), solver algorithm choice (Trilinos, built-in, naïve), and memory placement (KNL DRAM vs. on-package MCDRAM), they were able to get a deeper understanding of the energy usage behavior of each of the workloads and possible areas where energy-efficiency could be improved.

In particular, for the Sandia SPARC application (see Fig. 8, research staff identified significant opportunity to reduce the power usage of its solve phase by lowering p-state with negligible impact on performance. This resulted in a 25% energy efficiency improvement for the realistic input problem configuration that was evaluated.

The next steps are to modify SPARC to utilize the Power API [23] to change p-states at solve and assembly region phase boundaries. This will enable evaluation of dynamic power management and validate the results obtained using static p-state control.

D. MCDRAM

Run-to-run performance variability of HPC applications complicates the performance tuning, development and benchmarking on production HPC systems. Memory performance can be a cause of this, and the KNL-based Cray XC systems contain a new type of memory, MCDRAM, which introduces new sources of variability, such as cache mode effects. The ability to detect, assess, and attribute these conditions can help users to understand why they see variation in run-to-run performance of their applications and to mitigate it.

1) ALCF: While it is potentially impossible to completely alleviate performance variability on Cray XC40 KNL-based systems, due to the way their architectures are designed (their design goals are not always performance-focused, but can also be focused on cost-optimality or programmability), the Argonne Leadership Class Facility (ALCF) aims to investigate ways to contain the effects of variability. To that end, they have been investigating performance variation and measures in conjunction with performance counters with the goal of determining meaningful variables and value ranges for use in production conditions.

ALCF has designed a set of experiments to detect performance variability in which the MCDRAM cache-mode page faults are collected at the beginning and end of each run. Fig. 9 shows box plots of the per-node bandwidth values reported by the STREAM benchmark run on 50 KNL nodes of Theta in Flat-quad mode (top) and Cache-quad mode (bottom). (Note that the y-ranges are not the same in both figures). The STREAM benchmark is run with different working set sizes, and significant variation is seen with the cache-mode for all working set sizes. Low performing nodes, indicated by the outliers below the main box plots, were associated with counter values indicating higher MCDRAM page fault rates.

While KNL MCDRAM supports both flat and cache mode, providing flexibility to the programmer, employing the cache memory mode can come with a potential performance penalty. Not only is the overall memory bandwidth with cache mode lower than that which is possible with flat memory mode, but also it comes with an additional complexity of performance variability. The direct mapped nature of the cache-mode operation of MCDRAM is the root cause for this variability. While the Cray "zone sort" helps mitigate the impact of the problem to some extent, it does not help reduce the variability completely.

E. HSN

Network contention can be a cause of significant performance variation. How to detect contention, to quantify contention and its association with performance, and to identify the causes of contention are all open questions. Successful understanding and attribution of network congestion can be used to mitigate congestion through allocation and scheduling decisions.

1) HLRS: The High Performance Computing Center Stuttgart (HLRS) has been developing approaches for automatic detection of the sources and victims of network contention-induced performance variation.

HLRS and Cray have developed [24] an Apache Spark-based tool which analyzes system logs from the SMW in order to automatically identify applications impacted by network contention and possible applications causing the contention. Their approach detects applications with high variability in runtimes and considers these as possible *victims* of network contention-induced variability. It further extracts applications running concurrently with the affected victim runs. Frequently co-scheduled applications are extracted and promoted to the top of a potential *aggressor* list.

This approach has the benefit of automatically identifying potential victims and aggressors, as opposed to requiring manual log search. On their Cray XC40, Hazel Hen, their approach has identified three potential aggressor applications. They are currently working with a developer of a confirmed aggressor application to make the application 'less aggressive.'

Complexities in the job environment, such as multiple apruns from a single Moab submission, jobs running for a pre-allocated constant time, and short run times increase the difficulty of job extraction. In addition, application runtime may vary for reasons other than network contention (e.g., allocation or input deck). HLRS is thus extending its work to include global HSN performance metrics in its analysis. Identification and location of congestion in the system would reduce the dependence on job run time in detecting

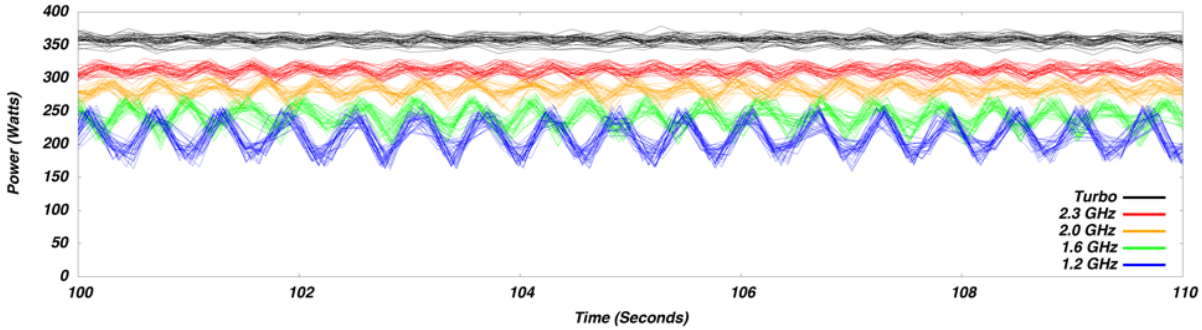

Figure 8. Zoomed point-in-time power profile of the SNL SPARC application running on Trinity Haswell compute nodes. Five runs are shown, each running with a different static p-state configuration shown in the key. The solve and assembly phases are more clearly visible as p-state is reduced.

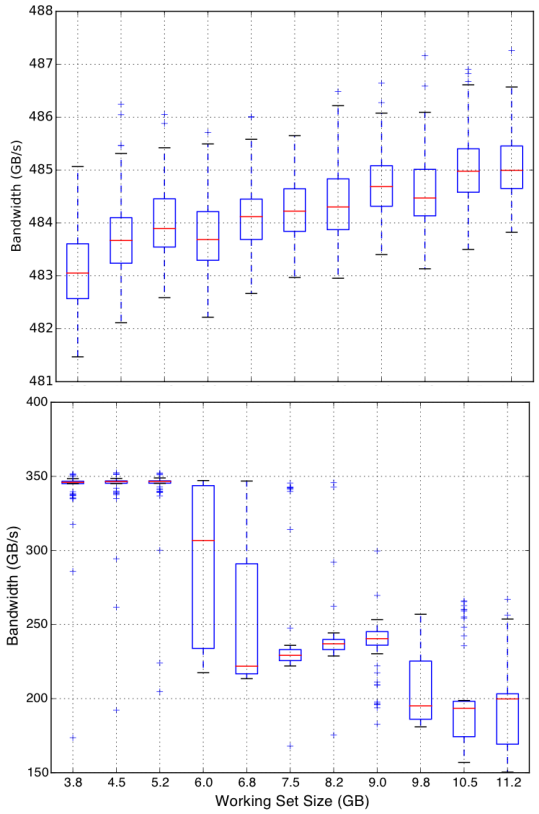

Figure 9. ALCF Performance variability experiments of the STREAM benchmark in Flat-quad mode (top) and Cache-quad mode (bottom) Significant variation is seen in the latter, for all working set sizes. Low performing nodes (individual marks below the main box plots for each case) have been associated with counter values indicating high cache miss rates.

performance-impacting contention and could enable better determination of the impacted jobs.

2) SNL: SNL and Cray have been collaborating to investigate the association of numerical measures of congestion with possible performance impacts. SNL has been collecting network performance counters via LDMS [25] at 1 second intervals on its Trinity Testbed System, Mutrino. Data is transferred off the system to a data store, where continuous analysis via NumPy is used to convert these counters into measures indicative of backpressures between the NIC and processor, the NIC and the HSN, and within the HSN. Details of the measures are beyond the scope of this paper, but include $stall/flit$ ratios at the various interfaces. NumPy was chosen because of its efficient computation on multi-dimensional arrays, which matches well with computations over components, variables, and time.

SNL is investigating congestion visualizations in both custom and commodity displays. Commodity displays are intended to provide at-a-glance understanding of the system-wide congestive state using statistical values of per-NIC and per-Aries backpressure; use of aggregate statistics ensures that commodity dashboard tools can provide run time performance for large component counts. Custom [26] displays provide more information in the context of the architecture.

A segment of a commodity display, via Grafana [27], is shown in Fig. 10. Percentages of the total system NICs experiencing various ranges of backpressure values from the HSN are shown in a stacked histogram. For this measure, a value of 1 roughly indicates that traffic is passing that interface at approximately half the rate that it would without congestion. In this case, application runs on 25% of the machine are experiencing potentially performance-impacting performance (shown in red, corresponding to values between 2-5). Such plots are being used to guide investigation of possible performance impact, using the detailed per-Aries, per-link values collected.

F. Environment

Operating specifications of a machine include requirements for cooling. These requirements are dependent on facilities conditions external to the Cray platform. While SEDC data is provided for the Cray cooling mechanisms, external facilities data is not innately integrated with that information in the Cray-provided monitoring data flow. Integrated monitoring can better help ensure correct operating conditions.

1) NERSC: During a period of record-breaking temperatures, NERSC needed to determine if a facilities response would be required to ensure that their two Cray systems continued to operate within safe operating specifications. The maximum inlet water temperature for safe operation is 75 degrees Fahrenheit, and external temperatures reached 92

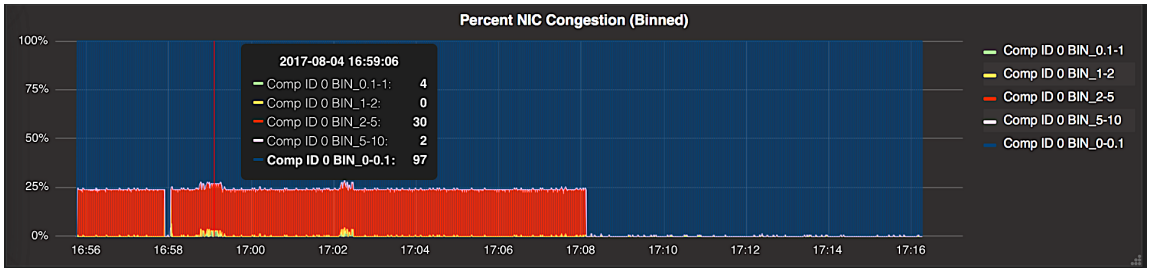

Figure 10. Stacked histogram of percentage of system NICs experiencing backpressure from the HSN on SNL system. Here, application runs on 25 percent of the machine (red) are potentially experiencing significant performance impact. ("Comp ID 0" in the display legend is an artifact referring to whole system information.)

degrees[1]. Potential facilities responses included a full system shutdown, but the far-less-disruptive option of reducing heat load was preferred, provided there was confidence that temperatures would remain in the safe operating envelope.

To determine if an external response was required, NERSC augmented their current system-level environmental monitoring (e.g., cabinet air and water temperature) to enable correlation with the facilities-level monitoring (e.g., pump speed, building water loop temperature and pressure) and with the external environmental conditions (e.g, outside temperature). They obtained the system SEDC data via Cray's off-ERD endpoint and forwarded it into their monitoring system. They obtained the facility data via a custom collector which read the facilities data stream as it was exposed in BACnet [28]. All data was integrated into common data streams and/or a common database for display in their Grafana dashboards.

Manual visual analysis allowed a gradual reduction of heat load by preventing new jobs from starting and, eventually, killing some running jobs, all without necessitating a full shutdown. Fig. 11 shows a mash-up of facility, environmental, and systems temperatures used by NERSC to monitor state over the course of the heatwave. During the event, they learned that their Cray systems, including cooling infrastructure, were more robust with respect to hot weather than was originally expected.

G. Trend Analysis

Trend analysis can reveal faulty components and system locations that are more prone to failure (such as those in hotter locations in the system) and enable predictive failure analysis. Trend analysis relies on the ability to collect, store, and analyze continuous system data.

1) ALCF: Environmental conditions and system and component errors are a fruitful source of data to support trend analysis. Currently, such data is exposed and collected on Cray systems, but in a limited way that does not easily enable analysis.

As described in Section III, SEDC data, transported over the ERD and collected into the PMDB on the SMW, is limited in its utility as a source for long term storage and computationally intensive analyses. Console and hardware error data, transmitted via the ERD, is filtered and dispersed into a variety of logs, some of which are in binary, which then require parsing to be suitable for use in analyses. As a result, ALCF needed a better way to obtain and integrate the data needed to give administrators a useful picture of the machine state.

ALCF has written a software stack [29], called Deluge, to listen to the ERD channels directly, bypassing the PMDB and binary log files, and make the data directly available for analysis. In writing Deluge, ALCF utilized Cray APIs and libraries, available in Cray development RPMs. These RPMs are not Cray supported resources, and thus are largely undocumented. In ALCF's environment, the data is sent to Elasticsearch or InfluxDB, with the data graphed and analyzed via Grafana or Kibana.

This work has proven invaluable to discovering and diagnosing various issues in real time on ALCF's Cray XC40 system, Theta. The hardware error data in Elasticsearch has been used to identify suspect hardware, and to discover trends in correctable/uncorrectable errors. Use of SEDC data has also proven to be operationally valuable, and ALCF's Influxdb install has proven to be faster and more storage-efficient than the PMDB, while the Grafana front-end has been used to identify and track datacenter environmental issues. As data in InfluxDB is kept forever, long-term trends can be tracked with a speed and simplicity not available through Cray's software tools.

One example of this analysis is ALCF's tracking of the BER (Bit error rate) of Aries links. This is defined as the ratio of the CRC mismatch packets to the total packets over the last minute on a per-link basis. Fig. 12 shows the BER for 3 links whose values are approaching the sustained error rate threshold of 1e-10, the threshold at which the Blade Controller software will disable links. Visualizations such as these are used to determine if there will be any instantaneous problems, what nodes might be affected, and which components should be replaced. Trends are examined, and this information can be used to highlight which Aries links are candidates for failure due to an excessive BER.

The component data used to calculate BER is part of the

[1]The previous day had reached 105 degrees, but was less critical due to a lower wet-bulb temperature, which is more important from a cooling perspective.

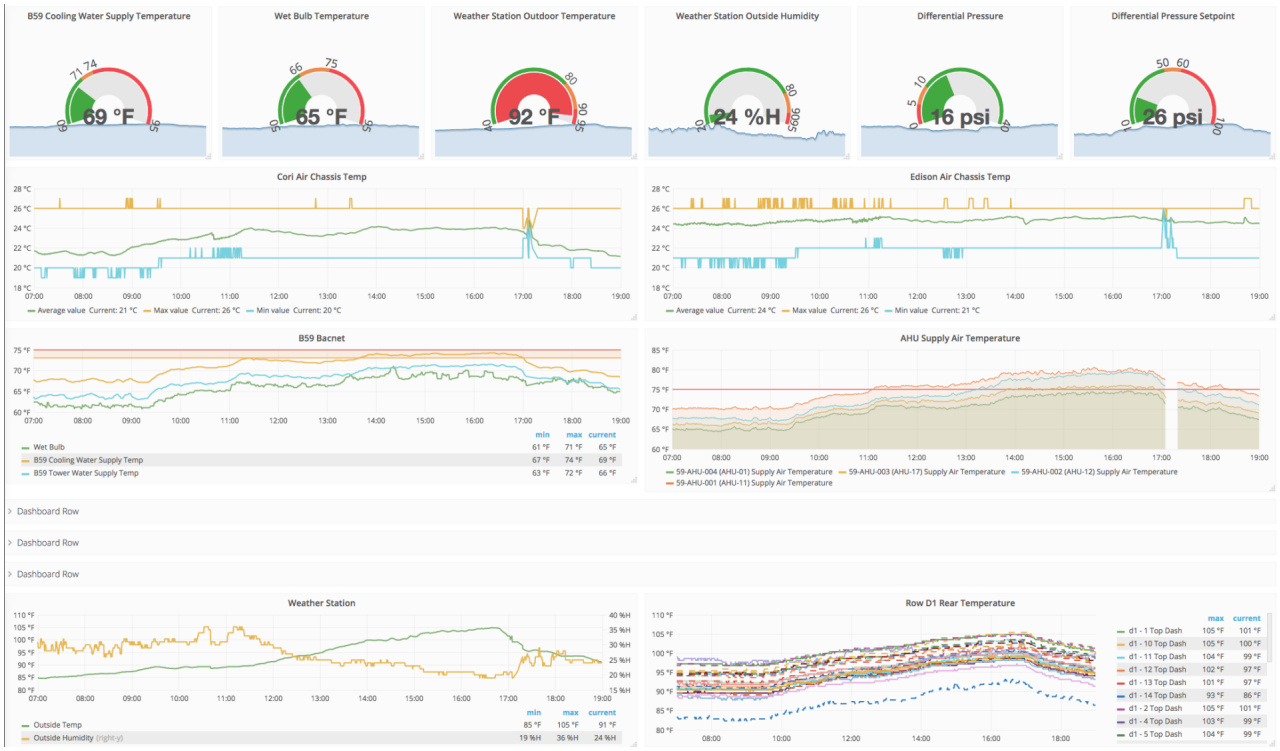

Figure 11. Systems, facilities and weather data in one dashboard (NERSC)

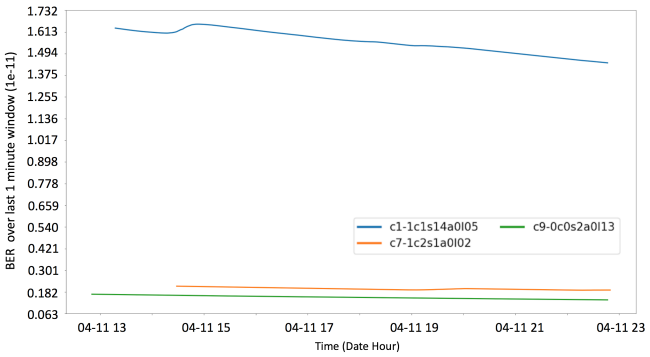

Figure 12. Trend analysis on ALCF system enables identification and replacement of potentially problem components before they are automatically disabled by the system and routed out (sustained Bit Error Rate of 1e-10).

hardware error data stored in Elasticsearch. A Python script grabs this data from Elasticsearch, and uses a typical data science stack (pandas and Matplotlib) to calculate the BER from the raw register data and graph it.

This work has shown that Cray's development RPMs are an underutilized resource, as developing directly against Cray's APIs has resulted in code that is high- performance and production ready, as opposed to reading text files or writing wrappers around CLI utilities. Cray-sanctioned support of such underlying capabilities, which would include the exposure and documentation of these APIs, would enable sites to more fully leverage these capabilities and build robust tools that could be shared between sites and make their systems more performant.

H. System Utilization and Queue Length Analysis

Understanding workload demands is key to understanding and improving throughput, making scheduling priority decisions, and determining future procurements.

1) CSC: The Center for Scientific Computing (CSC) has developed a graphical view for system queue length monitoring to visualize system utilization and obtain early indicators of system problems. CSC's system, Sisu, is configured for multiple partitions with different priorities. In order to get a high level view of the queue length over time, CSC has defined a theoretical measure '*Sisu day*' as all the core seconds queued to be computed scaled by the capacity of the system. If it would be possible to run all the pending jobs (queued and running) with 100% node utilization, this is how long it would take for all the jobs to complete. A single job utilizing the whole system with duration of 24 hours would equal queue size of one '*Sisu day*'. After running for 12 hours, there would be 0.5 '*Sisu days*' left.

Data for the analysis is gathered from Slurm via Nagios and presented graphically with Grafana. An example is shown in Fig. 13. Queue length and core utilization are shown. The 'gc' queue (yellow) starts to back up (around 2/19 evening) due to job submissions. Since this is a high priority queue, these submissions affect the throughput of other queues. Once a large job in the 'gc' queue finishes and its queue length drops (around 2/25 midday), CPU utilization drops as well, reflecting the now-free nodes. Visualizations such as this can inform users wanting to understand why their job wait varies and if there are large priority jobs

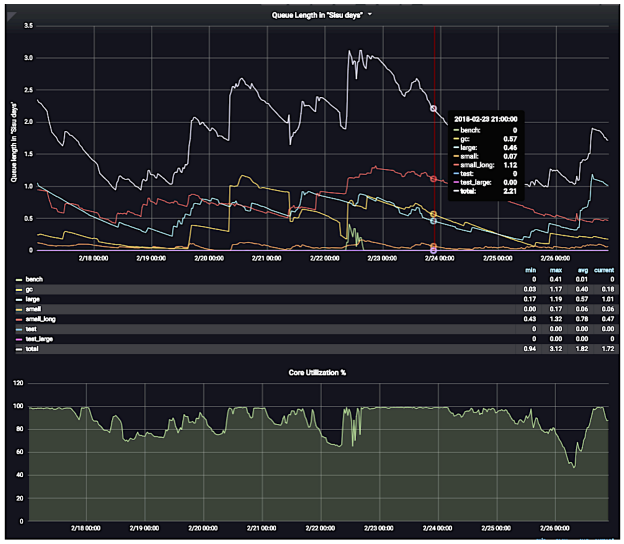

Figure 13. CSC Sisu job queue length development over 7 days. Occurrence of high priority jobs (yellow queue) and their effects on the throughput of other queues can be seen.

affecting the system usage.

System administrators and service desk staff use the graphical interface to determine differences from normal patterns of queue behavior and to drill down to see interesting time periods. Ultimately, CSC would like to combine the queue length measures with other monitoring information in order to detect and diagnose system problems, such as file system problems.

2) NERSC: NERSC characterizes jobs on the system in terms of node count, job duration, resource requirements, application, and other characteristics in order to fine-tune system configuration for maximum effectiveness and to support decision-making about future system requirements.

Data (including resource requests, timestamps relating to submission, eligibility to start, start and termination time, command-line details, etc.) is collected from Slurm about every submitted job and supplemented with other data collection points such as Altd [30], [31]. This data is stored in a database and used by web-based dashboards to compute and display various statistics, such as the mix of job size and the total volume of queued work.

Similar to CSC, NERSC monitors the *backlog*, that is, how many system-hours ($node-hours/num\_nodes\_total$) of work is in the batch queue. Detection of changes in the backlog has enabled NERSC to identify user submission issues. Fig. 14 shows the backlog over a period of time on the Haswell nodes of Cori. A sudden increase from the recent average of about 17 days to 85 days prompted closer inspection of the queue, revealing that a user had submitted a 1000-member array job, each member requesting 1000 nodes. Specifying 'total resources' instead of 'per-member resources' is an error users occasionally make when writing array jobs, so the user was contacted and alerted to the possible error. Shortly later, the user canceled the erroneously-

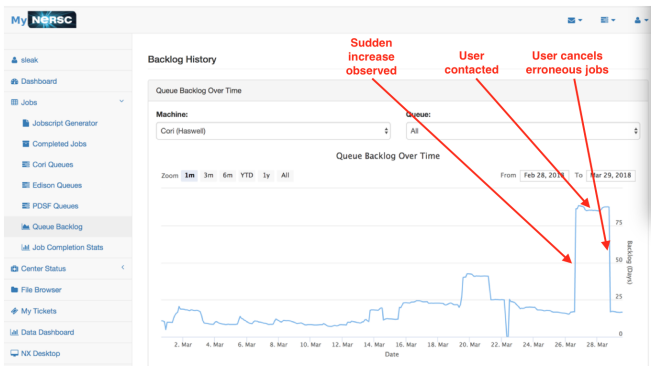

Figure 14. Backlog in the queue on NERSC Cori. Abnormal behavior can be used to detect problems.

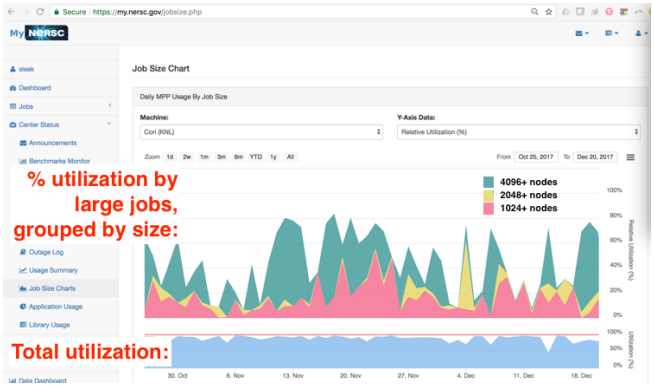

Figure 15. NERSC visualization facilitates assessment of operational job requirements. The 20% target large job utilization is clearly met.

specified jobs.

Large-scale system procurements are justified by the need to enable scientific investigations that require large resource counts. NERSC has an operational target that 20% of its KNL-node usage support jobs requiring at least 1024 nodes. Fig. 15 shows the overall utilization and the percentage used by jobs of certain sizes, filtered to highlight these large jobs. This gives a visual impression of how well they are tracking the 20% target.

I. Job Analysis and Reporting

The utilization of an HPC system and its subsystems can vary widely depending on what applications are running on the system. In order to have a clearer idea of the overall state and health of the system, it is important to include monitoring of job- and application-related state in evaluations. In addition to providing system and subsystem health-related insights, job/application monitoring can provide clarity into how users are using the system for running applications, and how applications can make more efficient use of system resources through code or configuration changes.

1) NCSA: On NCSAs 27,648 node Cray XE/XK Blue Waters system, monitoring infrastructure subsystems/components (collection, storage, alerting, job association, visualization) are intentionally decoupled from one an-

other behind the scenes for a number of reasons. The primary reason is to accommodate individual technology component evolution that minimizes encumbering interdependencies. However, it is still desirable to provide a user interface that couples data from all infrastructure subsystems/components for visualization and analysis of metrics, with options to scope views to subsets of those subsystems/components (e.g. all compute nodes, just compute nodes belonging to a job, just Lnet routers belonging to home filesystem, etc), as well as to scope the timeframe of interest.

In order to support this approach, NCSA actively monitors all major components of the Blue Waters system in a periodic and synchronized fashion in addition to passive asynchronous collection of all pertinent log messages that are generated on the system. Specifically, NCSA utilizes a combination of system wide performance state data collection and periodic probes (see Section IV-A2) of normal subsystem activities to identify abnormal latencies in order to aggregate sufficient information for performing productive system-level and job-level analysis and problem source determination.

Using this approach, viewing job-specific metrics is simply a matter of identifying the set of nodes assigned to a job and aggregating data (metrics) of interest from those nodes over the window of time the job was run. While starting data exploration from a job of interest is described in the case in this section, it is important to note that often when the system is behaving irregularly as a whole, an offending/suspect job must be identified without any hints. In such cases, data exploration typically begins with a system aggregate view of metrics plotted as a time series.

In the NCSA interface, individual points in time can be moused over, drilled down upon, or pivoted from in order to view the aggregate data point broken out into its constituent components (all nodes contributing at the clicked-on point in time). This methodology enables the user to quickly explore an anomaly in a time series by breaking it down (via a mouse click) to see if all nodes or just a few are contributing to it. Using the mouse-over feature at this stage now enables the user to see job information about contributor nodes of interest for that time or time interval, as they will usually stand out. Very frequently, contributors to an anomaly will have the same user or jobid in the mouseover information on the pivoted time series point. From there, another pivot can be performed (via mouse click), to display just the job specific data. Once a suspect job is identified, broader metric analyses focused on that particular job can begin. An actual example from the Blue Waters system follows.

A number of very large (22,000 nodes) job requests on Blue Waters were running for a very small fraction ($< 20\%$) of their requested time, and then exiting in error due to an out of memory condition caused by the application. This was resolved by the user, but subsequent runs were monitored by staff, a process which includes looking at metrics from jobs in question. Though the jobs were completing successfully, the staff had concerns over the jobs' efficiency. A sample job was selected for analysis and metrics revealed a couple of potential issues which were relayed to the user. Screenshots from the interface are shown in Fig. 16 and insights from the data are described here.

First, it shows that the job was targeting I/O at the home filesystem and not the performance-purposed scratch filesystem. I/O load on the home filesystem was light in this case, but the use of home for application I/O is discouraged by system policy. Second, there is a question of what the job is doing about 1/4 of the way into its runtime where its average aggregate load drops sharply (Fig. 16 (B)). In the same time frame, the job shows a small amount of network transmission (Fig. 16 (C)) and small average write volume to the home file system (Fig. 16 (E)). Note that the plots in Figs. 16 (C) and (E) appear to have the same periodicity and we can infer that the home filesystem writes are at least partially generating the observed network traffic. Unix Load (Fig. 16 (B)) that picks up after 10:00 roughly correlates with network traffic, home filesystem write volume, and reduction in node free memory. Also, the Unix Load correlates well with cache misses (Fig. 16 (D)) and FLOPs (Fig. 16 (F)). Put simply, it appears that the job is not doing much for a long stretch of runtime.

The user agreed that something looked terribly wrong after seeing the metric data, and took a closer look at the application. A few days later, the user indicated that they had solved the problem, realizing that they were using a parallelized eigensolver for large symmetric matrices on a very small matrix during one stage of their workflow which greatly expanded the runtime.

A high level summary of our approach is: Identify suspect job from aggregate metric exploration (this step was bypassed in this case since support staff were already shepherding a previously troubled workflow): Job was not using any resource heavily for a large portion of its runtime. Assessment by the support team: Able to quickly access job metrics for the jobs in question. Interact with the user about behavior and present data: The user, upon seeing the metric plots showing unexpected anomalies, was able to quickly identify the stage in the workflow where unexpected behavior occurred, and subsequently discovered and corrected an error in their job construction. The result was a realization that the targeted scale was too large for the problem size; the user *reduced* their job size to 8 thousand nodes and significantly shortened their runtime in the process.

V. Prioritizing Requirements

High performance computing systems are extremely elaborate, with multiple complex subsystems integrated and working in concert toward one end - performing science. As a user or member of the support team maintaining the system, the ability to understand the status, utilization, and

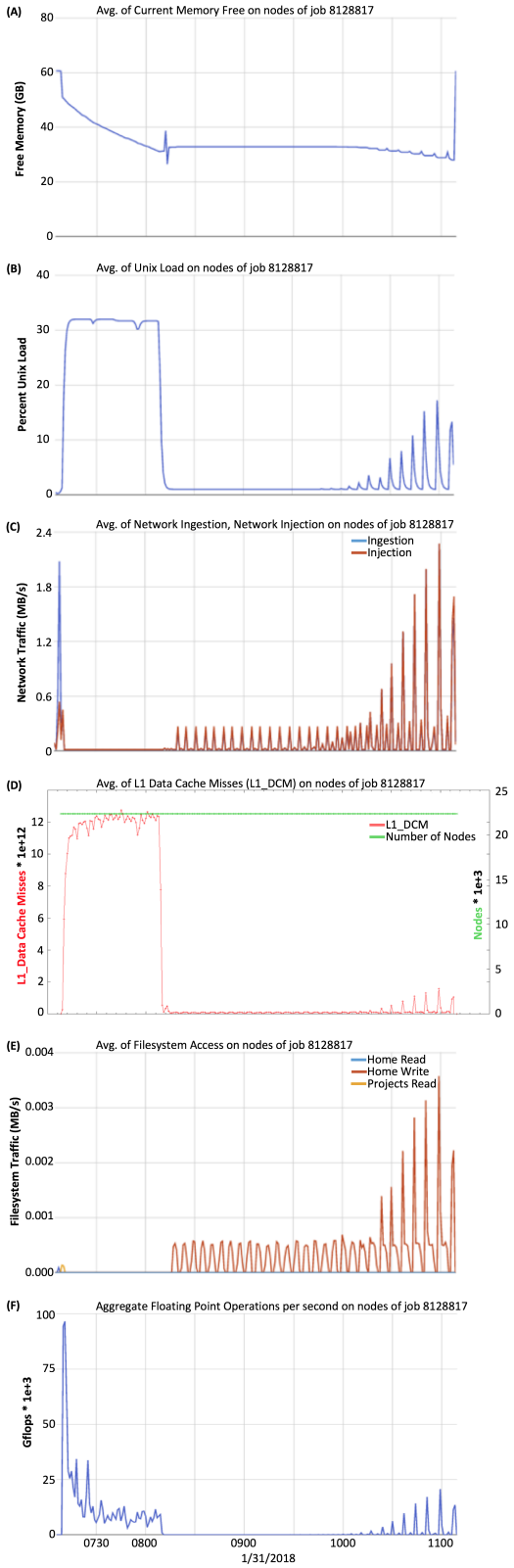

Figure 16. NCSA: job related aggregate datasets enable problem diagnosis.

performance of the system is key. Each vendor's solution uses different technologies, so it is unfeasible to expect every system to be capable of being monitored in exactly the same way.

However, there are steps that vendors should take to enable easier and more productive monitoring of their systems. First, and foremost, vendors should design systems where the ability to monitor is a *core capability* of the system. In many cases today, monitoring is added on as an afterthought, often by the sites themselves, and vendors have not designed the system to easily support this.

Systems to be monitored should to be designed with two goals in mind: (1) Make the data that the system generates readily available (customer can access) and (2) Make the data that the customer can access usable (the customer can interpret and present the data). Specific requirements to enable these two goals are given in Table I.

The monitoring cases in this paper all had to incorporate novel solutions to overcome difficulties in data availability or usability. In addition, a number of cases targeted getting fundamental information through active probing, non-Cray data collection and assessment, and even bypassing some Cray tools to get closer to the data streams.

Unfortunately, the current state of the data and data streams makes it difficult for sites to share solutions or even realize the full potential of their own solutions. Solutions that do not use Cray-supported data paths and interfaces risk breaking without warning when interfaces change. Cray site personnel have reported that they cannot use some site-developed diagnostics in their analysis because it does not enable a common diagnostic procedure when interfacing with Cray's support organization.

It would be of particular value for Cray to make the data available in officially supported ways and to provide the base set of capabilities, as described in Table I. The base set of capabilities is intended to support the general basic needs of all sites. Some level of extensibility is needed to ensure that reasonable site-specific customizations can be easily added and supported in the same web portal. Sites could also pull all of the data into their local monitoring solution and not rely heavily on the Cray-provided base solution, particularly for enacting more complex analysis. Note we cannot know in advance all the analyses that will be of interest, nor do we expect a vendor to provide all the analyses.

We recognize the difficulty for Cray and other vendors in providing and supporting a capability that is designed to be extensible. For example, the Node Health Checker (NHC) is easily extensible, with its ability to run site-defined scripts. However, the service database (SDB) is not; sites cannot add tables to the SDB to track other characteristics of the system outside of what Cray has configured in the SDB. The System Monitoring Working Group wants part of its collaborative interaction with vendors like Cray to be working jointly to answer questions like these so that Cray can design a

Table I
PRIORITY REQUIREMENTS FOR MONITORING

Make the data available	
Requirements	Notes
All data from all subsystems should be available through either APIs or in a format that is consistent, well-documented, and easily accessible (i.e. not binary-only)	Enable access to raw data where feasible
The data should be available in customizable subsets and frequencies (up to some limit defined by the vendor's technology)	Vendor to document possible performance impacts and allow the customer to do cost-benefit analysis, rather than pre-limiting data and/or rates
Enable exposure of the data to multiple consumers	e.g., users have access to information affecting their jobs, data can be directed to multiple analysis and integration consumers
Make the data usable	
Requirements	Notes
Provide a base set of capabilities that turn available information/data into actionable insights in an easy-to-consume way, such as a web portal with alerting capabilities	Basic required information: • Component status (e.g., compute nodes up/down/error status) • Subsystem performance/utilization information (e.g., filesystem metadata operations are at 37% of peak, HSN is congested in these areas) In support, vendor to provide *some* level of appropriate collection, transport, and storage mechanisms
Available data should be well documented - what it describes, and, if inference is needed to get a meaningful value, how to combine the individual data sources to get gain meaningful understanding	e.g., multiple network counters needed to calculate congestion on a port on the network
All subsystems across the system should be time synchronized, so that when multiple data sources are being combined for inference, clock skew is not adversely affecting the interpretation	

solution that is supportable, but meets the real needs of its customers in ways that it is not able to today.

VI. CONCLUSIONS

In this work, we have presented production examples of monitoring, the capabilities developed in order to enable them, and the insights gained from them. We have made priority recommendations to improve the ability to monitor systems. Understanding the potential benefits of monitoring and the improvements needed to achieve them can drive the changes necessary to enable more effective monitoring insights for code developers, users, system administrators, and system architects. Engagement of these stakeholders can help drive vendors to design monitoring as core capability like hardware, system software, and programming environments are today. Ultimately this can help ensure operational performance and guide future procurements.

ACKNOWLEDGMENT

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.

This document is approved for release under LA-UR-18-23035.

Contributions to this work were supported by the Swiss National Supercomputing Centre (CSCS).

This research used resources of the Supercomputing Core Lab of King Abdullah University of Science and Technology (KAUST).

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications.

This research used 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.

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility under Contract No. DE-AC05-00OR22725.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," *Official Journal L.*, vol. 119, no. 1, 2016.

[2] M. Showerman, A. Gentile, and J. Brandt, "Addressing the Challenges of Systems Monitoring Data Flows (BoF)," in *Proc. Cray Users Group*, 2016.

[3] Cray Inc., "Using and Configuring System Environment Data Collections (SEDC)," Cray Doc S-2491-7001, 2012.

[4] ——, "Monitoring and Managing Power Consumption on the the Cray XC30 System," Cray Doc S-0043-7003, 2013.

[5] ——, "Cray S-0043: Create a Remote Power Management Database," (Accessed 6.April.18). [Online]. Available: https://pubs.cray.com/content/S-0043/CLE\%206.0.UP06/xctm-series-power-management-and-sedc-administration-guide/create-a-remote-power-management-database#C320909

[6] ——, "intro_LLM man page," (Accessed 9.Jan.14).

[7] ——, "XC Series System Administration Guide (CLE 6.0UP06)," Cray Doc S-2393 Rev A, March 2018.

[8] NVIDIA, "NVIDIA System Management Interface," (Accessed 2018). [Online]. Available: https://developer.nvidia.com/nvidia-system-management-interface

[9] R. Alverson, D. Roweth, and L. Kaplan, "The Gemini System Interconnect," in *Proc. 2010 IEEE 18th Annual Symposium on High Performance Interconnects (HOTI)*, 2010.

[10] G. Faanes *et al.*, "Cray Cascade: A Scalable HPC System Based on a Dragonfly Network," in *Proc. Int'l Conf. on High Performance Computing, Networking, Storage and Analysis (SC)*, 2012.

[11] Cray Inc., "Aries Hardware Counters," Cray Doc S-0045-20, 2015.

[12] ——, "Performance Measurement and Analysis Tools," Cray Doc S-2376-63, 2015.

[13] ICL UT, "Performance Application Programming Interface," (Accessed 2018). [Online]. Available: http://icl.cs.utk.edu/papi/index.html

[14] Cray Inc., "Cray Linux Environment (CLE) 4.0 Software Release," Cray Doc S-2425-40, 2010.

[15] ——, "xtmemio man page," (Accessed 2018).

[16] ——, "CLE XC System Network Resiliency Guide," Cray Doc S-0041-E, 2015.

[17] Intel, "Intel xeon phi processor performance monitoring reference manual volume 2: Events," March 2017.

[18] Linux Programmer's Manual, "perf_event_open man page," (Accessed 2018).

[19] S. Alam, N. Bianchi, N. Cardo, M. Chesi, M. Gila, S. Gorini, M. Klein, C. McMurtie, M. Passerini, C. Ponti, and F. Verzelloni, "An Operational Perspective on a Hybrid and Heterogeneous Cray XC50 System," in *Proc. Cray Users Group*, 2017.

[20] Cray Inc., "CAPMC API Documentation," Cray Doc S-2553-10 Rev 1.0, 2015.

[21] NekBox, "NekBox," (Accessed 2018). [Online]. Available: https://github.com/NekBox/NekBox

[22] H. Ltaief, D. Gratadour, A. Chara, and E. Gendron, "Adaptive Optics Simulation for the World's Biggest Eye on Multicore Architectures with Multiple GPUs," *ACM Platform for Advanced Scientific Computing*, 2016.

[23] R. E. Grant, M. Levenhagen, S. L. Olivier, D. DeBonis, K. T. Pedretti, and J. H. L. III, "Standardizing power monitoring and control at exascale," *Computer*, vol. 49, no. 10, pp. 38–46, Oct 2016.

[24] D. Hoppe, M. Gienger, T. Bonisch, O. Shcherbakov, and D. Moise, "Towards Seamless Integration of Data Analytics into Existing HPC Infrastructures," in *Proc. Cray Users Group*, 2017.

[25] A. Agelastos *et al.*, "Lightweight Distributed Metric Service: A Scalable Infrastructure for Continuous Monitoring of Large Scale Computing Systems and Applications," in *Proc. Int'l Conf. for High Performance Storage, Networking, and Analysis (SC)*, 2014.

[26] J. Brandt, E. Froese, A. Gentile, L. Kaplan, B. Allan, and E. Walsh, "Network Performance Counter Monitoring and Analysis on the Cray XC Platform," in *Proc. Cray Users Group*, 2016.

[27] Grafana Labs, "Grafana," (Accessed 2018). [Online]. Available: http://grafana.com

[28] ASHRAE SSPC 135, "BACnet," (Accessed 2018). [Online]. Available: http://bacnet.org

[29] A. Kristiansen, "Use of the ERD for Administrative Monitoring of Theta," in *Proc. Cray Users Group*, 2018.

[30] M. Fahey, N. Jones, and B. Hadri, "The Automatic Library Tracking Database," in *Proc. Cray Users Group*, 2010.

[31] NERSC, "Altd," (Accessed 2018). [Online]. Available: http://www.nersc.gov/users/software/programming-libraries/altd/