Large-Scale System Monitoring Experiences and Recommendations
Workshop paper: HPCMASPA 2018

Ville Ahlgren[†], Stefan Andersson[§], Jim Brandt[x], Nicholas P. Cardo[‡], Sudheer Chunduri[*], Jeremy Enos[**], Parks Fields[‖], Ann Gentile[x], Richard Gerber[††], Michael Gienger[§], 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,(gienger|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*—Monitoring of High Performance Computing (HPC) platforms is critical to successful operations, can provide insights into performance-impacting conditions, and can inform methodologies for improving science throughput. However, monitoring systems are not generally considered core capabilities in system requirements specifications nor in vendor development strategies. In this paper we present work performed at a number of large-scale HPC sites towards developing monitoring capabilities that fill current gaps in ease of problem identification and root cause discovery. We also present our collective views, based on the experiences presented, on needs and requirements for enabling development by vendors or users of effective sharable end-to-end monitoring capabilities.**

***Keywords*-HPC monitoring; monitoring architecture; system administration**

I. Introduction

In machine acquisitions, monitoring is typically low-priority, if not an afterthought. While much attention is paid to potential component, sub-component, and sub-system performance (e.g., node, memory, filesystem), limited effort is made to ensure that information necessary to the continued operations and performance of the machines as a whole is made available in a meaningful way. The responsibility for identifying performance-impacting conditions at the operational level and in production thus falls on individual site's system administrators and depends upon the resources available to them. However, since application performance is affected by system conditions (e.g., components performing as expected, minimal network congestion), a lack of insight into such conditions may have significant impact on the production performance of a system. Furthermore, future platforms' specifications continue to be based on limited information of the actual production demands of sites' workloads; and the number and expertise of staff required to manage the platforms increases as the systems increase in size and complexity.

As a result, system administrators spend a significant amount of time developing tools to provide necessary in-

sights. However, they are hampered by unavailability of necessary data and insufficient understanding of the implications of system conditions on application performance. Such availability and understanding requires the active, continued participation of the vendor and component manufacturers who can enable the needed interfaces and can provide insight into the design.

In [1], a group of Cray sites presented monitoring capabilities they are developing which have resulted in improved understanding, performance, or operations efficiency. The goal of that work was that promotion of such value-added capabilities could drive the inclusion of monitoring as a core capability of HPC systems. In that work, we specifically addressed issues of data availability and usability that limited progress and should be priority items in future monitoring system requirements.

In this work, we extend the conclusions drawing on our current state of monitoring, including those stories; gaps that are impediments to additional progress; and our visionary use cases and goals of monitoring to address in detail needs and requirements for complete, end-to-end monitoring capabilities, encompassing data sources, storage, analysis, and visualization, and with accompanying architectural concerns. While the platforms of focus for this group and for the work presented are from a single vendor, the challenges, approaches, and a number of the tools are generally applicable to any HPC system.

The remainder of this paper is organized as follows: We present site-specific approaches to priority issues being addressed by monitoring in Section II. We discuss the spectrum of approaches, as they address gaps and goals in monitoring in Section III and Section IV. Throughout, we have highlighted in bold, key issues and takeaways. Finally we discuss insights, needs and requirements based on the presented approaches and experiences, to enable development of sharable effective tools for large scale HPC monitoring, analysis, and response in Section V.

II. Identifying sub-optimal operation at large Cray sites

The sites participating in this work run large Cray systems and have independently developed mechanisms to help ensure that user jobs run reliably and predictably on their systems. The range of monitoring targets is wide since a single run of an application may occupy thousands of nodes and utilize tens of thousands of components across several functional subsystems (e.g., storage, network). Anomalous behavior in a component or subsystem often indicates a problem in either some part of the system or in the application or job specification and all of these sites monitor for anomalies according to perceived or previously-experienced sources of sub-optimal operation.

The efforts summarized below appear on the surface to be site-specific approaches to identification of site-specific problems (e.g., GPU performance, shared network performance, shared filesystem performance) in operating large scale HPC systems. Closer observation, however, reveals that the differences are largely in the particular implementation details and/or particular components or subsystems being targeted while the types of problems are generally the same across sites. More detail of the efforts and targets can be found in [1]).

1) Los Alamos National Laboratories (LANL): Trinity is a 20,000 node Cray XC40 system sited at LANL. To **detect anomalies as early as possible**, LANL staff have developed a suite of custom tests that are performed, system-wide, on 10 minute intervals across all relevant components and subsystems including but not limited to: configurations (e.g. on burst buffer nodes); verification that essential services and daemons are functional, including filesystem mounts; and ensuring there is an appropriate amount of free memory on compute nodes as well as available space in shared filesystems. The results of the tests are displayed in a dashboard which also presents significant log data occurrences, in order to facilitate problem diagnosis.

2) National Center for Supercomputing Applications (NCSA): Blue Waters is a 27,648 node Cray XE/XK system sited at NCSA. In order to **facilitate efficient use of system resources, early detection of poor performance in critical shared resources, comparison of performance metrics over time (see Figure 1), and to minimize time to solution when troubleshooting**, NCSA actively collects data from all major components and subsystems of Blue Waters at one minute intervals. Collection times are synchronized across the entire system. Additionally NCSA passively collects all pertinent log messages generated on the system as they asynchronously occur.

Performance problems in any of the three large shared Lustre file systems can severely impact job performance and system efficiency. Thus NCSA staff have additionally developed a set of probes that execute on one minute intervals and measure file I/O and metadata action response latencies. These target each independent filesystem component and run from a distributed set of clients to exercise these operations over representative data paths from all applicable software environments. This provides a view into the current and historical experience that the applications and interactive users have when interacting with the filesystem.

3) National Energy Research Scientific Computing Center (NERSC): NERSC hosts Edison and Cori, which are 5,586-node and 12,076-node Cray XC systems, respectively. Like LANL, NERSC regularly runs a suite of custom benchmarks that exercise compute, network, and I/O functionality, and publishes performance over time on its user-facing web pages (illustrated in Figure 2). These enable staff to **identify abnormal or unsatisfactory behaviors in these subsystems** and take remedial action - which might include further analysis to determine the root cause of the anomaly.

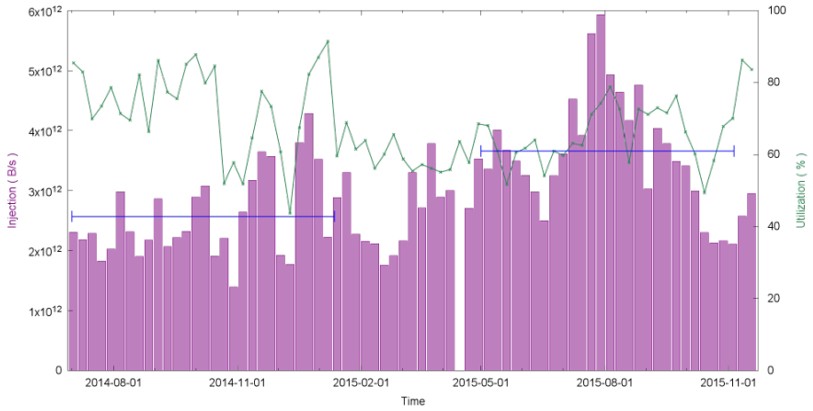

Figure 1. NCSA: Use of system wide continuous collection of performance data (injection of data into the network in this case) to enable comparison of resource utilization between two periods of time where there were different scheduling and resource allocation strategies in effect. Shared network resources are being utilized more after Topologically-Aware Scheduling (TAS) [2] was put into production (right) than before (left). This is shown by the blue line representing mean bandwidth utilization as a percent of maximum and which is significantly lower over the pre-TAS time period (left) than when TAS was being utilized (right).

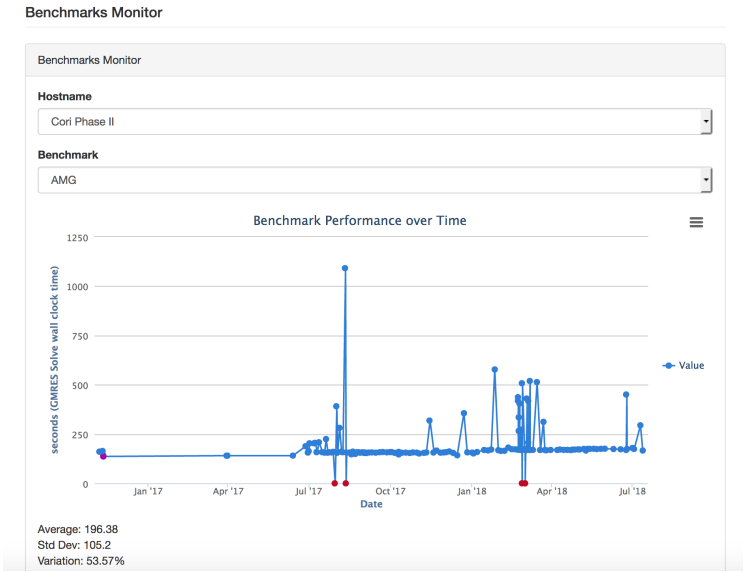

Figure 2. NERSC: Custom benchmarks are regularly run to assess functionality and performance. Occurrences and onset of performance problems are apparent in visualizations tracking performance over time and are used by staff to drive further investigation and diagnosis.

NERSC also monitors the batch queue backlog - large or sudden changes in outstanding demand can indicate for example a spike in jobs that fail immediately upon starting (quickly emptying the queue) or a blockage in the queue (quickly filling it).

Finally, NERSC captures large volumes of environmental data about its systems and facilities, both for real-time operational monitoring (power use, spare cooling capacity) and for post-hoc research (e.g., one might correlate power usage against applications).

4) Center for Scientific Computing (CSC): Sisu is a 1,688 node Cray XC40 system sited at CSC. Like NERSC, CSC utilizes queue length monitoring and display, but in order **to provide users a realistic view into the expected wait time** for the currently submitted workload. CSC plans to utilize characterization of queue length together with other monitoring information to help identify and diagnose system issues such as shared file system problems.

5) Swiss National Supercomputer Center (CSCS): Piz Daint, sited at CSCS, consists of 5,320 XC50 nodes with NVIDIA GPUs and 1,813 XC40 nodes without GPUs. In order **to validate the health of the GPUs in their systems**, CSCS also has developed a comprehensive suite of tests. Their philosophy is that "no 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." In this case the test suite is run before and after each job. If the pre-job health assessment fails another node is chosen and the problem node taken out of service for further testing and possible repair. If the

post-job health assessment fails, the problem node is taken out of service for further testing and possible repair.

6) Oak Ridge National Laboratory (ORNL): Titan is a 18,688 node Cray XK7 system, sited at ORNL. Around 2.5 years into its production deployment ORNL began to see an increasing rate of GPU failures. Following a lengthy process of tracking failures and performing a variety of in-depth component level diagnostics on the failed GPUs, it was determined that NVIDIAs manufacturing process for the SXM had not used sulfur-resistant materials. The result was growth of crystalline structures which modified the resistance characteristics of a small number of resistors and caused malfunction. **To ensure new and replacement hardware is free of this issue**, ORNL now monitors their data center environment to ensure that ASHRAE standards for particulate and corrosive gases are exceeded and enforces the use of sulfur-resistant materials in their supplier's Bills of Material.

7) King Abdullah University of Science and Technology (KAUST): Shaheen2 is a 6,174 node Cray XC40 located at KAUST. The KAUST staff take a different approach to monitoring for abnormal behavior on Shaheen2: **to identify poor performance**, rather than devising direct tests of subsystems, KAUST utilizes power monitoring to accomplish the same goal. Their power monitoring was originally intended to enable them to understand the power requirements of different applications and to stay within a particular power budget while executing their workloads. However, they found the power profiles of applications were repeatable enough that they can, through profiling, characterization, continuous monitoring, and comparison against power profiles of known good application runs, identify problems with the system and applications. Anomalous power-use behaviors within a job can also be used to detect problems such as hung nodes or load imbalance (e.g., Figure 3).

8) Argonne Leadership Class Facility (ALCF): Theta is a 4,392 node Cray XC40 system sited in the ALCF. ALCF staff have written a tool called Deluge [3] to obtain data from Cray's Event Router Daemon [4] (ERD) and make it directly available for analysis. ALCF is utilizing this information to **gain critical understanding of system state**. Information available from the ERD includes environmental data, and both console and hardware error data. ALCF currently performs trend analysis, using this data, on component error rates (e.g., High Speed Network (HSN) link Bit Error Rates (BER)) and the datacenter environmental conditions. Based on these trends, ALCF personnel can flag and diagnose unusual behaviors on component and subsystem levels.

9) Sandia National Laboratories (SNL): Researchers at SNL in collaboration with Cray engineers and several Cray sites have been investigating the use of functional combinations of High Speed Network (HSN) performance counters [5], collected periodically (1 - 60 second intervals) and synchronously across a whole system, **to determine**

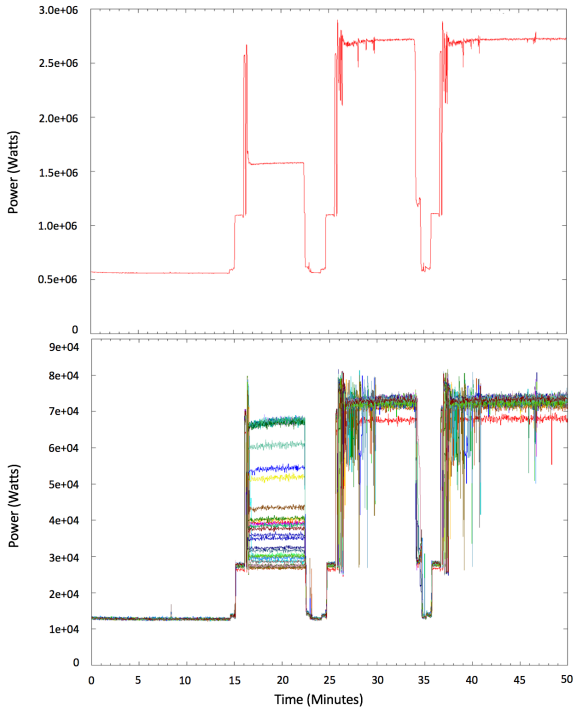

Figure 3. KAUST: Shaheen2 overall power usage (top); power usage per cabinet (bottom) (from [1]). Load imbalance issue was detected with power usage variation. Around 17-22 minutes, power usage variation of up to 3 times was observed between different cabinets and full system power draw was almost 1.9 times lower during this period of variable cabinet usage. This was rectified for subsequent runs.

congestion levels, congestion regions, and impact on application performance. This work targets the Cray Aries-based dragonfly networks and Gemini-based 3D torus, with work under way to apply their approach more generally to other technologies and topologies.

SNL, like KAUST, also investigates power profiling, sweeping configuration parameters such as p-state, power cap, node type, solver algorithm choice, and memory placement, with the goal of **improving application and system energy efficiency while maintaining performance targets**.

10) High Performance Computing Center Stuttgart (HLRS): On their 7,712 node Cray XC40, Hazel Hen, HLRS has worked with Cray to develop an approach for **identifying "aggressor" and "victim" applications based on their runtime variability**. Applications having high runtime variability are classified as "victim" applications and those running concurrently that don't hit the "victim" variability threshold are considered as possible "aggressor" applications where the resource being contended for is assumed to be the HSN.

III. Spectrum of Approaches - Analysis

In this section and the next we address key aspects of the monitoring approaches taken by the sites.

Here, we identify commonality in the goals and approaches to analysis, visualization, and response taken by the

participating sites. We also describe common themes in the challenges sites face when implementing these approaches. These themes represent opportunities for vendors to enable easier to use and more portable analysis and visualization tools and to facilitate acting on the insights gained.

A. Data Sources – A Summary of Approaches

Three mechanisms are used by sites to identify poor performing components, including applications and system configuration:

- CSCS, KAUST, LANL, NCSA, and NERSC have crafted benchmark suites whose constituent benchmarks exercise specific components in the system (e.g., computational capability, memory bus bandwidth, file system meta-data operations) and whose performance metrics (e.g., time to solution, energy consumption) enable identification of poor performance of those features.
- ALCF, NCSA, ORNL, KAUST, and SNL also utilize periodic or job launch related (pre-, post-) job collection of performance counters and state registers, and asynchronous hardware and console errors. These can be obtained from a variety of sources including the `/proc` and `/sys` file systems; the Performance API [6] (PAPI); Model-Specific Registers (MSRs); network performance counters; Cray's ERD, System Environment Data Collections [7] (SEDC), and Power Management Database [8] (PMDB); and GPU specific diagnostic tools (e.g., [9]).
- Information from the job scheduler or resource manager, especially log files and queue depth information, enables CSC and NERSC staff to identify anomalous queue depth behavior which may be associated with poorly performing components such as filesystems and networks. This data source also enables HLRS operations staff to identify possible occurrences, causes, and victims of HSN contention via information on concurrently running applications and execution times.

B. Analysis and Visualization

Commonalities in analyses used at different sites and the challenges addressed by each site to obtain them reveal where vendors can add the most value to their monitoring capabilities.

Currently most analyses implemented by sites seek to **assess current component state**. This is fundamental for operations and might be expected to be a basic production functionality, but in each case **sites have needed to develop tests and infrastructure to enable continuous testing, reporting, and response**.

Sites have long been interested in early detection and, ultimately, prediction of component degradation and failure based on trend and outlier analysis. This has had limited success in production; **better ability to track component data at rates that capture events of interest**, and, through vendor interaction, **increased domain knowledge of expected component behavior and performance interdependencies** may increase success in such areas.

Log analysis has significant research history involving techniques of abnormality detection and/or variation in occurrences of log lines. However, **in production most log analysis involves detection of well-known log lines**. Detection of conditions of interest and the best way to represent them is a continuous process; new events and new signatures for events may arise with new workflows which stress the system in new ways, or with new versions of system software which may present new or different log messages. Thus, new or infrequent events may be missed until manual observation of events leads to identification of relevant log lines to include in the scan.

Events that propagate over components are especially complex and might span long time periods - for example, delays in recovery from HSN link failures may impact other components using the HSN. These require a vendor-supported understanding of the architecture and system mechanisms along with an ability to associate log lines across time periods.

Associating numerical or log events over components and time is particularly tricky when **a single global timestamp is unavailable** as local clock drift can result in erroneous associations.

Lack of **access to comprehensive data** or log lines may result in missed events or attributions; however analysis of such large data may require distributed storage and analysis capabilities for timely processing.

Users frequently request an explanation for observed performance variation. Understanding and attributing this variation has been reported [10] to be the highest priority question sites seek to answer and is the motivator for many current monitoring efforts at sites. Periodically running benchmarks assists in detecting performance variation but greater interaction with vendors is sought to **better correlate variation with relevant system metrics** - for example identifying components and conditions that might impact application performance and, if so, to what extent.

Interestingly, sites may have the first large-scale installations of architectures, where issues may then first arise, while vendors may only have limited access to local small-scale systems for investigating the interplay of platforms, system software, and applications. While investigation of some essential issues (e.g., slow boot times, complex job launch with Burst Buffer support) has been done on large site installations as part of stand up and acceptance, sites are increasingly interested in **offering site resources for collaboration with the vendor** in investigating at-scale performance and operational improvements, such as network and filesystem contention performance impact analysis.

Dashboards for visualization of status are a common

practice across sites. Grafana [11] is currently a popular first order solution, due to its ease of configuration, ability to graph live data, and ability to copy and share dashboard configurations. However, **individual component graphs may decrease in value and performance** as the number of components plotted increases. Representations in the context of the architecture, such as network-topology representations, are being developed by sites and others (e.g. [12], [13]) for these architectures, however visualization of complex connectivities is a challenge. Reduced dimensionality through higher-level aggregations (e.g., percentage of components in a state, regardless of location) coupled with drill-down capabilities can enable better at-a-glance understanding.

Figure 4 shows how NCSA has used an aggregate (read bytes/sec for an entire filesystem) visualization over time to provide a direction for further investigation, coupled with drill-down to show, at a selected time, the metric across components.

NCSA also provides the ability to download both plot images and the associated Comma Separated Value (CSV) formatted data (e.g., Figure 5) to enable **controlled release of data to users**. Even with per-job aggregation, per-metric plots take up substantial screen real estate. Per-job analysis requires storing and extraction of job allocations and timeframes, which adds to storage and query complexity. NCSA uses Google Charts [14], assembled via PHP, to visualize data in real time. These appear to be stable in handling visualization of Blue Waters scale of data and present no critical server- and client-side dependencies.

C. Response

Systems, their components, and system software typically provide some level of detection and response capabilities, such as thermal throttling and re-routing around failed network components. For more general cases and extensible use, vendor-provided or widely available tools such as Cray's Simple Event Correlator [15] (SEC), Splunk [16] and Nagios [17] enable response when well-known conditions are met, typically via regular-expression matching. Responses are typically simple - such as issuing an alert or marking a node as down - and the detection and associated response(s) has been developed by sites.

More complex responses to more sophisticated conditions of interest are envisioned. **Notification to users of assessments of system conditions** is of interest but relies on the proper analysis. **Scheduling and allocation based on application and resource state** is an active area of interest. Topologically-aware scheduling based on largely-static routing rules has been implemented in production (e.g., [2]). Power-aware scheduling seems likely to become important with increasing scale. More generally, sites would like the capability to perform more fine-grained and dynamic resource allocations and task mappings. This would require

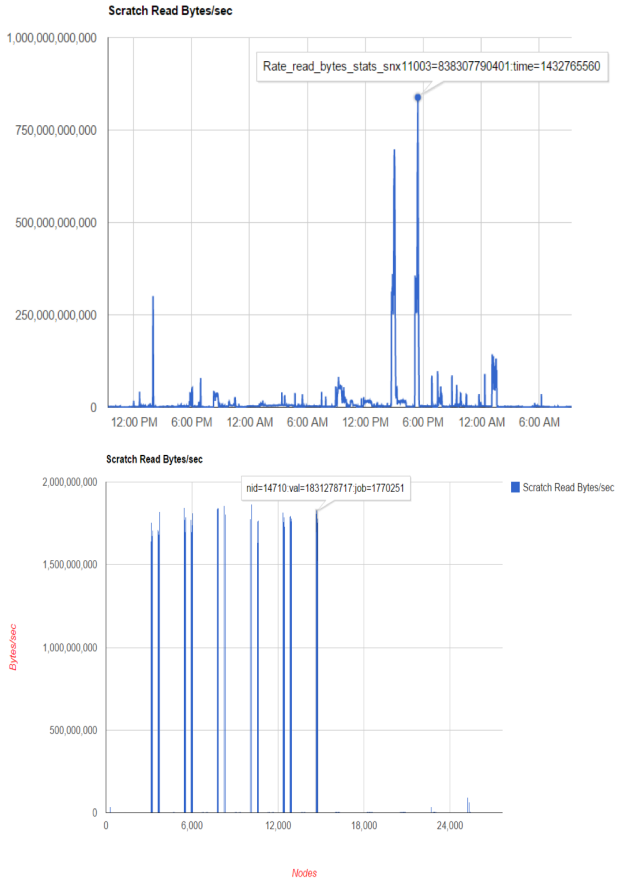

Figure 4. NCSA: Drill down capabilities enable investigation while limiting screen real-estate requirements. Here high values of system aggregate I/O metrics (top) drives further investigation into the nodes, and hence, the job responsible for the I/O.

increased analysis capabilities and **more complex interfaces to schedulers and component/subsystem controls** (e.g., downclocking components).

Clear potential financial gain may increase the priority for development of such capabilities – for example, sites envision the redirection of power between platforms and even between other site resources based on both current and anticipated needs.

IV. Spectrum of Approaches - Architecture

In this section we identify commonality in the goals and approaches to monitoring used by the participating sites in terms of architectural issues such as transport and data storage. We describe common themes in challenges that the sites face when implementing monitoring solutions. These themes represent opportunities for vendors to enable more portable and extensible system monitoring, and ultimately more efficient operation of machines.

A. Use of undocumented and unsupported data paths to obtain data

Several sites have utilized **unpublished and/or unsupported codes and APIs** for collecting system data for use in

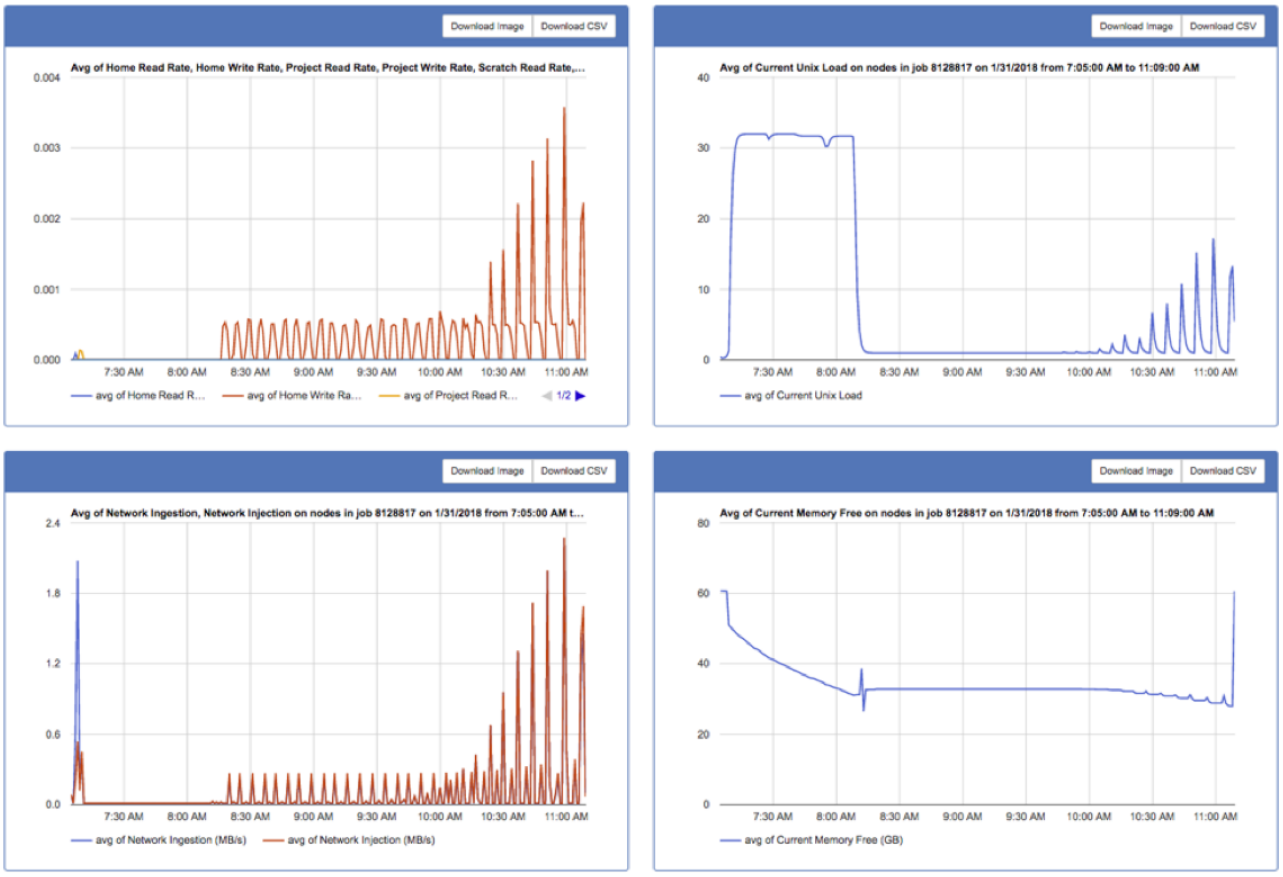

Figure 5. NCSA: Timeseries visualizations of multiple metrics can provide insights into underperforming applications. Summing and averaging over nodes enables condensation of high dimensional data enabling at-a-glance understanding. NCSA enables user access to plots, with the ability to download the image and also the raw data for further investigation. Infrastructure to support data collection, transport, storage, analysis, and visualization is not supplied by the vendor.

assessing health and/or identifying problems and associated root causes. This includes data that is either not exposed, or exposed by methods which may be difficult or inefficient to use for the intended analyses.

Vendors selling integrated solutions are positioned to develop instrumentation and APIs for acquiring data from instrumented components. Unfortunately, **vendors often develop proprietary solutions that are meant to provide telemetry for fundamental system operation but are not meant to be utilized by operations staff as data sources**. In Cray's case, they acquire a vast amount of data which is 1) transported in a proprietary binary format (a small subset is made available to operations staff in text format for troubleshooting purposes) or 2) used exclusively internal to the system for detection of vendor defined problem conditions and to trigger mitigating responses. These two cases appear in our site experiences only because site personnel, through personal interaction with Cray engineering staff and/or reading through source code, became aware of these sources and how to access them. With respect to case 1, ALCF personnel utilized source RPMs to understand what data was flowing from Cray's Event Router Daemon (ERD) (which transports all event information), how to access the raw encoded data, and how to decode the data. In case 2, SNL personnel were made aware of the libraries and APIs to enable querying the Gemini and Aries performance counter data and what functional combinations of which data could be used to provide insight into network congestion levels.

While these interactions have been fruitful for particular sites, problems arise when collaborators wish to leverage the non-Cray developers' tools at other sites. The libraries and interfaces are not officially opened to everyone (some even falling under NDA agreements) and there is no guarantee that the APIs won't change in future system software releases. Thus, these kinds of tools will necessarily remain one-off tools and every site will continue independent development of their own monitoring capabilities. If vendors would provide **in depth documentation of such capabilities and user accessible APIs for reading data** that is already being produced, sites could collaboratively develop and share tools across sites. This could also enable abstractions that would facilitate sharing across multi-vendor platforms.

ALCF's case also illustrates that **vendor transla-**

tion/filtration of data may result in less usable forms of data. By default, Cray separates log events into at least 20 different per-day log files, addressing different sources and/or types of events (e.g., hardware errors, network events), and placed into a multi-level directory hierarchy. Time and date formatting vary between files, some log events are multi-line, and some files are binary requiring Cray tools to provide human-readable translations. It is possible to forward the log stream off the system and thus bypass some of the formatting and separation. Generally, however, data translations, formatting, and filtering then require significant parsing to identify and combine the underlying data. ALCF's work enables them to get data closer to the source and to its native format, enabling more usable and complete data from the ERD event stream.

B. Data Transport For Storage and Analysis

Whether monitoring data is obtained via benchmarking, querying through standard and/or supported APIs, or via some other mechanism, the data still needs to be transported from the source and put in stable storage for processing. Currently data transport and storage (Section IV-C) are handled on a per-data-type basis where the only standard is use of some version of *syslog* for transport of log (e.g., error and event) messages.

Benchmark execution times, and, optionally, other embedded figures of performance, are handled on a per-site basis. Typically, these are first stored to the running user's specified location and then used for analysis and visualization. In this case the shared filesystem is the transport and storage is first performed by writing output files which are then harvested for pertinent information which is finally put into some kind of site-specific store (e.g., file, database) for analysis and comparison against expected results.

For other types of data, sites utilize a variety of transport mechanisms, possibly concurrently, including vendor proprietary (e.g., Cray's ERD), custom built (e.g., SNL's Lightweight Distributed Metric Service (LDMS) [18]), standards based (e.g., Redfish [19]), popular (e.g., AMQP [20], RabbitMQ [21]), etc. Of all the transports listed here, the only proprietary and platform dependent one is Cray's ERD. While this offers hope for the process of tool sharing across platforms, sharing across sites is made difficult because **different sites have made different choices among the plethora of available data transport and related storage mechanisms**. These choices reflect a variety of things such as data type, variety, and fidelity and associated performance requirements; directly applicable analysis tools; and supported visualization tools including dashboard displays. As such, multiple transports may in some cases be necessary and even desirable. Currently, however Cray provides *no* **generally accessible transport mechanism and storage solution**, even for data limited in time and size (although the PMDB has been a move in this direction). As a result, sites must make a substantial up-front effort in design and implementation for fundamental monitoring capabilities and which might not then be a drop-in at other sites.

C. Data storage and formats

While information/data being collected (described in Section II) is similar as are the analysis goals, sites have picked a variety of data storage technologies for storage and analysis.

- *MySQL* [22] NCSA stores compute-node performance data in a pre-existing MySQL database containing other system and workload data. NERSC uses MySQL for a variety of job, software usage and node-state data. These decisions are motivated primarily by convenience of querying the variety of data and work well syntactically, but canonical implementations of SQL-based databases lack scalability with respect to ingest, deletion, and query impacts and performance.
- *Power Management Database (PMDB)* [8] Cray's PMDB is a PostgreSQL [23] database with a schema developed by Cray to support primarily power data and which also supports the SEDC data. Originally designed to be located on the System Management Workstation (SMW), it now can be stored separately via ERD (Event Router Daemon) forwarding capabilities [24]. Recently-added functionality allows job-log data to be forwarded to the PMDB, which can be used for extracting per-job data. As with MySQL, PostgreSQL suffers from performance limitations. Also, to support additional data sources PMDB needs schemas and other supporting capabilities (e.g., analysis and visualization) to be developed. A recent OS update (UP06) uses TimescaleDB - also based on PostgreSQL - to improve performance [25].
- *Elasticsearch* [26] NERSC and ALCF use the ElasticSearch ecosystem for storing and analyzing monitoring data. NERSC's infrastructure includes a message queuing system (RabbitMQ) and incorporates data from building management systems and weather stations as well as its clusters, while ALCF also stores job data and uses its custom tool to read hardware error (*hwerr*) data from the ERD. Both use LogStash to format data from disparate sources appropriately for ElasticSearch.
- *Splunk* [16] and *Simple Event Correlator* (SEC) [15]. Several sites use Splunk for their log message analysis, which consists of developing regular expressions matching messages of interest and utilizing Splunk's visualization interface for examining occurrence data. These sites format their test suite output to support indexing with Splunk which can then visualize test suite results in conjunction with log data. This is not a storage format but requires storage for the indexing of the log files in their native format. Splunk's pricing model is based on size of the data being indexed, which

Table I
NEEDS AND REQUIREMENTS FOR MONITORING

Needs	Requirements
Architecture	
• We will always need additional data. We will always need to use it in unanticipated ways. We will always need higher fidelity data. • We will need to direct the data and analysis results to multiple consumers (e.g, users and administrators, various analysis capabilities). • We will need to integrate the data with other non-platform data.	• Vendors should provide well-documented interfaces for accessing raw data at maximum fidelity with the lowest possible overhead. • Platform owners should be able to determine the data access, transport, storage, and performance tradeoffs of their own choosing, rather than vendors limiting accesses or amounts. The monitoring system, both hardware and software, should be provisioned with this in mind, with options for scaling up. Where access and transport of data might incur impact, that impact should be well-documented. • Multiple flexible data paths should be anticipated, with changes in data direction and data access easily configured and changed. • All monitoring system capabilities should be production capabilities and documented, exposed, and supported as such. • Tools to transport and store the data in native format are highly desirable. • Extensibility and modularity are fundamental to support evolutionary development.
Data Sources	
• We will need production-level insight to the compute platform and all supporting subsystems. This should reveal state of health, utilization, performance, and performance-impacting events. • We will always need more raw information and we will need information that leverages domain knowledge the platform supplier has about its own architecture and about other vendor supplied components.	• Potential data sources include traditional text (e.g., logs), numeric (e.g., counters) sources, as well as test results and application performance information. • Vendors should expose all possible data sources for all possible subsystems. • The meaning of all raw data should be provided. Computations required to extract meaningful quantities from raw data should be defined. • Continuing interaction throughout the lifetime of the machine with vendor engineering staff is needed for understanding the data in the context of the architecture.
Data Storage and Formats	
• We will need to keep all data and to analyze historical data in conjunction with new data in unanticipated ways.	• Easy access to historical data and the ability to access historical data in conjunction with current data is required. As with the storage of application data, all storage does not have to be equally performant; hierarchical storage models with the ability to locate and reload data as needed are desirable. • Analysis results should be able to be stored with raw data.
Analysis and Visualization	
• We don't expect the platform supplier to provide all analytics, nor to know all the analytics that we might need now and in the future. We will need to develop investigatory analyses and visualizations. • We will want to support both immediate feedback and post-processing analyses. • We will want to assess application performance in conjunction with system performance and utilization measures.	• Analysis capabilities should be supported at variety of locations within the monitoring infrastructure (e.g., at data sources, as streaming analysis, at the store, at points of exposure to consumers). • The data store should be designed to support arbitrary extractions and computations. Stores with interfaces that support popular computational packages are desirable. • Concurrent conditions on disparate components should be able to be identified. • High dimensional and long term data need to be handled in analyses and visualizations. • Visualization interfaces and tools should facilitate easy development of live data dashboards.
Response	
• We will need to take action on the results of the analysis. This includes feedback to both humans and software.	• Reporting and alerting capabilities should be easily configurable. These should be able to be be triggered based on arbitrary locations in the data and analysis pathways. • Data and analysis results should be able to be exposed to applications and system software.

can be cost-prohibitive during event storms such as when problem conditions arise on the machine.

Cray systems more generally use SEC, which can trigger events, such as alerts, upon matching conditions. This is also not a storage format but a processing tool.

- *InfluxDB* [27], ALCF reads SEDC data from the ERD and inserts it to InfluxDB in real time. ALCF retains the data indefinitely and uses Grafana for visualization. InfluxDB was chosen for its superior data compression and query performance for high-volume time series data

compared to Cray's PMDB, though the recent move of PMDB to TimescaleDB has prompted a re-evaluation of this decision.

Sites are interested in **long-term data analysis**, which may require revisiting historical data in conjunction with current data and may involve applying new analyses to historical data. As a result, storage methodologies which enable keeping near-term data in performant storage, and complete long term data in perhaps less-performant storage which can be reloaded into active data are desired. Solutions must address both the mechanics of the **archiving and reloading and tracking** the locations and contents of archived data.

Estimates of data sizes and ingestion rates have historically been generally lacking, based on log size and only vendor-obtained numeric data. As sites increase both their data collection, including external data (e.g., facilities data) to be integrated, and analysis results to be stored, these size estimates and formats will need to be revised.

Storage must also support efficient analysis. Many historical monitoring data storage solutions in the HPC space have *not* been designed to support the amount of data, its fidelity, the possible complexities of queries, and the modern analysis tools (e.g., [28], machine learning) needed for current and envisioned uses of the data. **Understanding of the intended queries, analysis, and analysis tools is essential for appropriate design**.

V. Conclusions

Based on our experiences and insights presented in Section III and Section IV, we have aggregated and formalized the key needs and recommendations in Table I. These are organized by components required for comprehensive production monitoring solutions.

In addition, our shared experiences across ten large-scale HPC sites have generated the following high-level insights into current critical gaps in vendor solutions with respect to monitoring, storage, analysis, and visualization tools and tool development:

- Component (hardware/software) performance metrics are not well-documented, not published, or both. This includes architecture-dependent processor and network performance counters; and includes both access methods and how they can be used/combined to provide insight into resource state and utilization.
- Interfaces for making large-scale data generally available are hard and enabling data exploration is even harder.
- Currently available storage (database) technologies do not lend themselves to the wide variety of use cases for aggregation and analysis of information (combination of event, text, numeric time series) with respect to capacity, performance, and size.
- Vendor tools don't generally integrate well with user developed tools across platforms.
- Vendor engineers in many cases don't have answers for system behaviors and performance-impacting issues and large-scale collaborative experiments may be the only path to discovery. Such experiments are costly and must be engineered thoughtfully before execution.
- In many cases site-specific researchers and operations staff work with vendor engineers via back channel relationships to acquire undocumented information which they then use to build one-off tools. These typically cannot be shared for a variety of reasons including bilateral NDAs and site reluctance to maintain and/or tools based on fragile or unsupported vendor code with no guarantee of being carried forward.
- Site-developed capabilities may not even be fully utilized at the sites themselves, since Cray on-site personnel may not be able to utilize individual site-developed diagnostics if they don't enable a general, global diagnostic procedure.
- Tools are often developed by/for administrators with root access and ubiquitous "need to know". Adding infrastructure to control information access per user is often impractical and hence information that might be of tremendous benefit in answering users' burning question(s) cannot be shared with them.

Successful monitoring will increase the demand for monitoring. Potential consumers of data will increase in number and type (e.g., system administrators, users, application developers, site platform procurers, intelligent system software, etc.) as data becomes more useful. Thus, end-to-end monitoring development will be a continuous, evolutionary process, and extensibility and modularily of all monitoring components' designs will be essential.

Acknowledgment

This research was supported by and used resources of the Argonne Leadership Computing Facility, which is a U.S. Department of Energy Office of Science User Facility operated under contract DE-AC02-06CH11357.

This document is approved for release under LA-UR-18-26485.

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.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number 2015-02674.

REFERENCES

[1] V. Ahlgren, S. Andersson, J. Brandt, N. P. Cardo, S. Chunduri, J. Enos, P. Fields, A. Gentile, R. Gerber, J. Greenseid, A. Greiner, B. Hadri, Y. H. He, D. Hoppe, U. Kaila, K. Kelly, M. Klein, A. Kristiansen, S. Leak, M. Mason, K. Pedretti, J.-G. Piccinali, J. Repik, J. Rogers, S. Salminen, M. Showerman, C. Whitney, and J. Williams, "Cray System Monitoring: Successes, Requirements, Priorities," in *Proc. Cray Users Group*, 2018.

[2] J. Enos, "Application Runtime Consistency and Performance Challenges on a Shared 3D Torus." MoabCon 2014, 2014. [Online]. Available: http://www.youtube.com/watch?v=FR274JitRq8

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

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

[5] ——, "Aries Hardware Counters," Cray Doc S-0045-20, 2015.

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

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

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

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

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

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

[12] 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.

[13] Y. Livnat, P.-T. Bremer *et al.*, "DragonView." [Online]. Available: https://github.com/LLNL/DragonView

[14] Google, Inc., "Google charts," (Accessed 2018). [Online]. Available: https://developers.google.com/chart/

[15] Risto Vaarandi, "SEC - simple event correlator," (Accessed 2018). [Online]. Available: https://simple-evcorr.github.io

[16] Splunk, Inc., "Splunk," (Accessed 2018). [Online]. Available: https://www.splunk.com

[17] Nagios developers, "Nagios," (Accessed 2018). [Online]. Available: https://www.nagios.com

[18] A. Agelastos, B. Allan, J. Brandt, P. Cassella, J. Enos, J. Fullop, A. Gentile, S. Monk, N. Naksinehaboon, J. Ogden, M. Rajan, M. Showerman, J. Stevenson, N. Taerat, and T. Tucker, "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.

[19] Distributed Management Task Force, Inc., "Redfish API," (Accessed 2018). [Online]. Available: https://www.dmtf.org/standards/redfish

[20] OASIS, "AMQP: Advanced Message Queuing Protocol," (Accessed 2018). [Online]. Available: https://www.amqp.org

[21] Pivotal Software, Inc., "RabbitMQ - Messaging that just works," (Accessed 2018). [Online]. Available: https://www.rabbitmq.com

[22] Oracle Corp, "Mysql," (Accessed 2018). [Online]. Available: https://www.mysql.com

[23] The PostgreSQL Global Development Group, "Postgresql," (Accessed 2018). [Online]. Available: https://www.postgresql.org

[24] Cray Inc., "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

[25] S. Martin, G. Koprowski, and S. Wallace, "Cray XC Advanced Power Management Updates," in *Proc. Cray Users Group*, 2018.

[26] Elasticsearch BV, "Elasticsearch," (Accessed 2018). [Online]. Available: https://www.elastic.co/products/elasticsearch

[27] InfluxData, Inc., "Influxdb," (Accessed 2018). [Online]. Available: https://www.influxdata.com/time-series-platform/influxdb/

[28] SciPy developers, "SciPy," (Accessed 2018). [Online]. Available: http://scipy.org