Enabling Advanced Operational Analysis Through Multi-Subsystem Data Integration on Trinity

J. Brandt*, D. DeBonis*, A. Gentile*, J. Lujan†, C. Martin†,
D. Martinez*, S. Olivier*, K. Pedretti*, N. Taerat‡, and R. Velarde†
*Sandia National Laboratories
Albuquerque, NM
Email:(brandt|ddeboni|gentile|davmart|slolivi|ktpedre)@sandia.gov
†Los Alamos National Laboratory
Los Alamos, NM
Email:(jewel|c_martin|ronv)@lanl.gov
‡Open Grid Computing
Austin, TX
Email:narate@ogc.us

Operations management of the New Mexico Alliance for Computing at Extreme Scale (ACES) (a collaboration between Los Alamos National Laboratory and Sandia National Laboratories) Trinity platform will rely on data from a variety of sources including System Environment Data Collections (SEDC); node level information, such as high speed network (HSN) performance counters and high fidelity energy measurements; scheduler/resource manager; and plant environmental facilities. The water-cooled Cray XC platform requires a cohesive way to manage both the facility infrastructure and the platform due to several critical dependencies. We present preliminary results from analysis of integrated data on the Trinity Application Readiness Testbed (ART) systems as they pertain to enabling advanced operational analysis through the understanding of operational behaviors, relationships, and outliers.

***Keywords*-High Performance Computing; Monitoring**

I. Introduction

At the same time HPC platform scale is increasing, systems are also becoming more heterogeneous in computational, storage, and networking technologies. As the volume and complexity of information continues to increase it will become impossible to efficiently manage platforms without tools that perform run-time analysis continuously on *all* available data and take appropriate action with respect to problem resolution and power management. An example of the complex interplay that advanced analysis tools can be used to aid in understanding is variation in application performance due to network congestion, contention for shared parallel file system bandwidth, contention for burst buffer bandwidth, thermally related CPU throttling, working but faulty hardware/firmware, and more. Any or all of the above conditions can cause wide variation in application performance but without simultaneous access to monitored facilities data, system data, system logs, console logs, event logs, etc., correct diagnosis and problem resolution will become impossible.

Additionally, large scale HPC platforms are now pushing the limits of data center power and cooling infrastructure. Modern large scale platforms with power draw requirements in the 20MW range can stress data center and site power infrastructure (e.g., power demands that change abruptly can cause power disruption in the data center and possibly including the local power grid). Thus the ability to prioritize and manage platform power allocations is becoming essential and active management of a platform's average and peak power draw through processor frequency management (another parameter that affects performance) has become a high priority for both HPC data centers and vendors and is currently a hot research topic.

Relatedly, the increase in power density of HPC components has necessitated the use of water based solutions for heat transport rather than traditional air cooling solutions. This in turn requires feedback mechanisms to maintain proper water temperature, pressure, and flow rates as well as active fan control in the case of hybrid solutions.

Traditionally our HPC platforms, both current and past, have had stove-piped monitoring operations where information rarely crossed the boundaries of responsibility between facilities and platform operation. Data centers monitored power and cooling at a plant level and HPC system administrators monitored platform level variables such as system and event logs. Only when the platform environment was deemed to be the cause of problems would there be communications

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.
LA-UR-15-23103

between the two groups about their respective data and how it was pertinent. However, upcoming pre- and exascale platforms will have the potential to incur large monetary cost and even cause site wide disruption to power if operated blindly.

Thus, in order to maximize the value of modern large scale platforms a management approach that tightly integrates all information both internal and external to the platforms is required. The ability to control data center infrastructure dynamically based on platform, job, power, and environmental information will become a necessity as we move towards exascale computing.

In this paper we present how we are planning on performing such information integration to efficiently manage Trinity, the upcoming ACES Cray XC40 platform. We show how we are utilizing our Application Readiness Testbed (ART) systems to prototype and validate our planned configuration with a particular eye to power, thermal, and data center facilities data. We first present our ART system and monitoring configurations in Section II. Next we describe the various sources of information along with how they are being collected/aggregated in Section III. In order to produce data under normal operating conditions we put together a workload to run on the system which is described in Section IV. Select data along with analysis of its pertinence is presented in Section V. Finally we present conclusions and future work plans in Section VI.

II. System Configuration and Monitoring Setup

Trinitite and Mutrino are ART systems obtained to prepare for Trinity with respect to the applications that will be run. Additionally, they provide platforms for validation of facilities–platform interaction and comparison. Most of the testing and measurements presented in this paper were performed on Trinitite, a single cabinet Cray XC40 at Los Alamos National Laboratory (LANL), where Trinity will also be sited. We also present some comparisons between behaviors of Trinitite and an identical system, Mutrino, sited at Sandia National Laboratories (SNL), in Section V. The ART systems were delivered in February 2015.

In this section we first describe the basic configuration of our ART systems. We provide insight, at a high level, about our integrated monitoring configuration, what information we are collecting, and our information aggregation and processing approaches. Finally we discuss the mechanisms used for transport of data from the various sources to a common system for processing. Note that the monitoring infrastructure on the full Trinity system will be distributed.

A. System Configuration

Our ART systems are single cabinet water cooled Cray XC40's populated with 100 compute nodes and 18 service nodes (the rack is not fully populated with server blades). The service nodes have the following functionalities: 1 logins, 6 burst buffer, 2 MOMs, 2 DVS, 3 LNet routers, 2 sdb (1 failover) and 2 boot (1 failover). Additionally linux white boxes are provided for: external login, System Management Workstation (SMW), and Power Management Node (PMN). The PMN is currently being utilized as a *Monitor* host. All compute nodes and service nodes utilize the Cray Aries interconnect in a dragonfly configuration [1]. Each compute node is configured with dual 16 core Intel Haswell processors running at 2.3GHz and 64GB of memory. The external connections consist of FDR Infiniband to Lustre storage (Sonexion), and 40/10 Gigabit Ethernet to SMW, sdb, external login, and PMN. The public network connections are 10 Gigabit Ethernet.

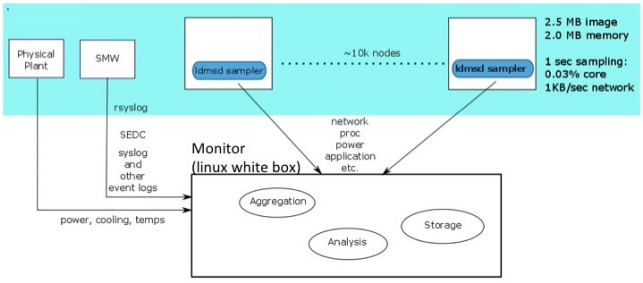


Figure 1. High Level Monitoring Diagram showing ART related information sources and their data feeds to a *Monitor* host

B. Monitoring Setup

Figure 1 depicts the current HPC platform components in the colored band with all monitored data, including that from the data center infrastructure, being sent/forwarded to a *Monitor* host deployed specifically for data aggregation, analysis, and both short and long term storage of the data. This scenario will change with the full Trinity system in that there will be several *Monitor* hosts for both scalability and redundancy with the 10K nodes depicted in the figure being an upper bound on what a *Monitor* host would be expected to process. As shown all monitored data will be aggregated to a set of *Monitor* hosts which will serve as data aggregation, storage, and both run-time analysis and post processing. This is the approach we have taken with data gathering for this paper where the *Monitor* box here is represented by the *PMN* block in Figure 2. The exceptions were that our data center infrastructure monitoring was performed out-of-band as described in Section III-E and we supplemented the SEDC and power data as described in Sections III-F and III-C.

Node level data collection is performed by Lightweight Distributed Metric Service (LDMS) samplers (Section III-D) running on every host (including login) as shown in Figure 2. Aggregators for this information run on *service* nodes (a login node in this case) and collect data at regular time intervals using Remote Direct Memory Access (RDMA) to minimize compute node CPU overhead. Aggregators

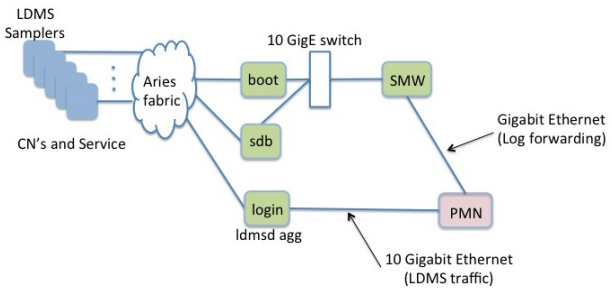


Figure 2. Application Readiness Testbed (ART) Connectivity Diagram showing how the major components of the platforms are interconnected

running on the *Monitor* hosts collect this data from the aggregators running on the *service* nodes and store the data. The aggregators running on the *Monitor* hosts are also capable of doing analysis on the data as it is streaming through and will ultimately be able to provide notification of outlier behavior. Note that the out-of-band Hardware Supervisory System (HSS) network exists but is not shown. All log files and SEDC data are forwarded to the *Monitor* from the SMW. On the full Trinity platform each *Monitor* will be receiving a fraction of the SEDC data with appropriate redundancy for failure mitigation.

III. Data Sources

In this work we utilize data from a variety of sources including System Environment Data Collections (SEDC), node level information, scheduler/resource manager, and data center environmental sensors. The SEDC data provides information about voltages, currents, and temperatures of a variety of components at the cabinet, blade, and node level. This data also includes dew point, humidity and air velocity information. While the system utilizes many of these measurements to identify out of spec, and hence unhealthy, components it relies on fixed thresholds being crossed to trigger knowledge of an unhealthy situation. The node level information provides high fidelity energy measurements, OS level counters, and high speed network performance counters. Scheduler/resource manager information provides time windows and components associated with user applications. Data center environmental data provides fine grained power draw, information about noise on the power feeds, and water temperatures and flow rates.

A. Logs

Logging when errors or meaningful transitions occur is used by many subsystems as a means of providing system administrators and troubleshooters with diagnostic information. On the Cray XC system many sources of log information exist: syslog, console, power_management, smw, event, Application Level Placement Scheduler (ALPS), etc. All of these logs are forwarded from various components to the SMW and placed in appropriate directories. In order to make them available for analysis in conjunction with the rest of the data we are collecting, we forward all log files to our *Monitor* host using *rsyslog*. This data path is depicted in Figure 1.

B. System Environmental Data Collections (SEDC)

Cray's System Environment Data Collections (SEDC) [2] provides a rich source of environmental data for many low level system components such as CPUs, memory, power supplies, nodes, blades, and more. The beauty of this information is that it is completely out-of-band with respect to node level computation and network communication. While the typical (default) configuration is to push the SEDC data to an aggregation point (the SMW) over the Hardware Supervisory System (HSS) network at 60 second intervals, we configured it to be pushed at 1 second intervals and configured *rsyslog* on the SMW to forward it to our *Monitor* host. One of the problems we see with the current configuration is that independent of platform size, all SEDC data is configured to go directly to the SMW. This presents a bottleneck to storage and parallel processing of this data. Ultimately we would like the ability to incorporate other devices, such as our *Monitor* hosts, into the HSS network and have the SEDC data distributed across them.

Some of the data that should have been available via the SEDC data stream was missing i.e., all cooling water related data, such as temperatures, pressures, and flow rates, was not present. Additionally it should be noted that while CPU Package energies units are defined as Joules in the SEDC scanID file, the actual data appears to be in units of milli-Joules (mJ). We did not use this data for comparison purposes in this paper as we could not validate that it was actually calibrated in mJ. Cray [3] states these bugs are fixed in CLE7.2 UP03 and CLE7.2 UP04 respectively.

C. Power API

We use the Power API prototype, which is a reference implementation of the Power API specification version 1.0[4] released to the HPC community in September of 2014, to collect node level data at 10Hz. The Power API specification describes a comprehensive system software API for interfacing with power measurement and control hardware. The specification defines the system model, theory of operations, and features exposed, covering the facility level down to low level software / hardware interfaces. The prototype supports most core features of the Power API specification. The prototype is a layered architecture that conforms to the specification and provides rich descriptive system configuration semantics, supports runtime plugins for a variety of devices and resources, and enables distributed communication for remote invocations of capabilities (see Figure 3).

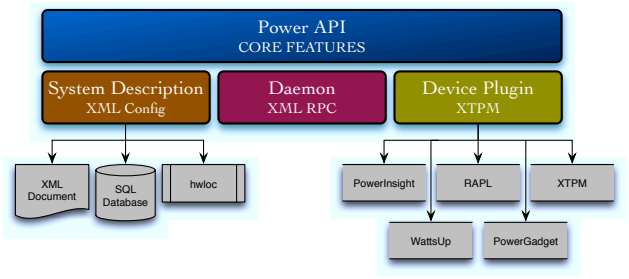


Figure 3. Framework of the Power API Prototype

For our study, we describe our system using a node level XML configuration file and utilize a plugin specifically created for the Cray platform which gains us access to the power management features of the system. The combination of the configuration file and XTPM plugin allow us to abstract the mechanisms of measurement and control from the specifics of the platform by mapping Power API attributes to the underlying plugin *sysfs* exposed parameters. We gathered data at a sample rate of 10Hz for each node of the system using this facility. In this case data was saved to a shared file system for post processing. In the future we will use LDMS (Section III-D) to transport this data directly to the *Monitor* for run-time processing.

Note that while Cray provides power monitoring capabilities and a power management database (PMDB) for storing and querying power utilization data, these are inadequate for our purposes. The power monitoring capability collects at lower frequencies than we are interested in investigating, and even when higher frequency data can be obtained, it is limited in its ability to handle large numbers of nodes and long time periods [5]. Additionally, the power database is located on the SMW, which has inherent limitations in access and size, while we seek to enable continuous, near-indefinite runtime and historical analysis integrated with other data sources. For these reasons, we do not include the power management database as source of data in this work. For convenience, however, we do use RUR output for some general, relative, overall application energy utilization.

D. Lightweight Distributed Metric Service (LDMS)

We use LDMS [6] for node level data collection and transport via the High Speed Network (HSN). This information, collected at 1Hz, includes HSN performance counters, Lustre client activity, application memory utilization, and other counters exposed by the OS. As of this writing LDMS collects node level energy data from the *sysfs* interface at 1Hz. The LDMS energy sampler plugin will be upgraded to collect 10Hz power/energy data via the Power API prototype (see Section III-C) thus making this full fidelity (10Hz) data available at 1Hz across the whole system. Use of the Power API will enable platform independent development of LDMS power data collection plugins that can take advantage of new power related features as they are developed without necessitating rewriting of the LDMS plugins.

The High Speed Network (HSN) performance counter data is exposed via Cray's *gpcdr* kernel module through the *sysfs* interface. While this module has been utilized for over a year on NCSA's Blue Waters platform (Cray XE/XK), this is our first use of it on a Cray XC. This exposed a small problem with the default *gpcdr* configuration which exposed 160 counters via a single *sysfs* file. While this configuration is ok when the counter values are small, it causes the aggregate size to exceed the 4KB limit imposed on *sysfs* entries as the values become large. The author at Cray quickly diagnosed the root cause of our apparent counter corruption and provided us with a simple fix which was to divide the initial set into 4 smaller sets based on information type (traffic, stalls, receive link status, and send link status) [7]. This required only a slight modification of the *gpcdr* configuration file and a reload of the kernel module.

We additionally collected the following non-HSN data via LDMS:

- Lustre file system counters
- CPU load averages
- Current free memory
- LNet traffic counters
- ipogif counters
- power and energy metrics via *sysfs*

E. Facilities

The facilities infrastructure that provides cooling to Trinitite is composed of two main loops (Figure 4): the primary and secondary cooling loops. These are separated by heat exchangers. The primary loop consists of four cooling towers and three pumps. The secondary loop consists of the three heat exchangers and three pumps on variable frequency drives (VFD).

The facility water supply temperature is 45^oF at the inlet to the heat exchangers. The secondary loop water supply temperature to the machine and preconditioner is 75^oF.

The building automation system that controls and monitors the facility cooling equipment is manufactured by Trane [8]. The Trinitite system and associated facility infrastructure installation was substantially completed in February 2015 and testing began in March 2015. Not all of the building automation systems were operational for this testing time frame.

The facilities data collected for this work was limited to power. Facilities power data was collected at five second intervals from the compute rack feeder breaker using a Fluke meter 1730 Energy Logger. The Fluke meter was set to sample at 5KHz. It calculates an average over a five second interval as well as capturing the minimum and maximum values over that interval.

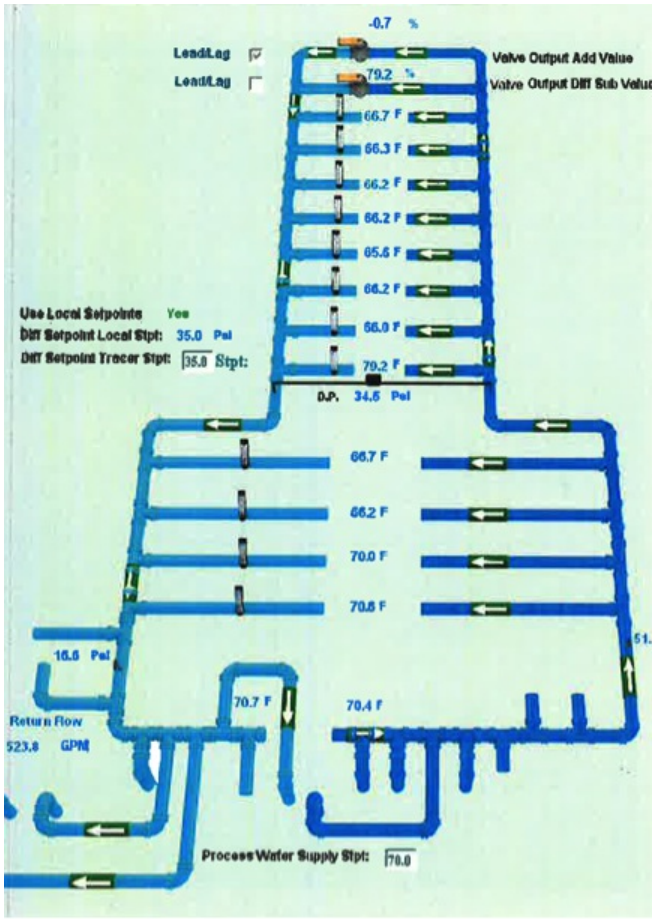


Figure 4. The Underfloor Pipe Diagram for the Data Center that houses Trinitite.

F. Envdata Script

Because the SEDC data stream currently does not contain the majority of the cooling water related data (see Section III-B) we have augmented our data collection using the envdata [9] script. From the SMW this script collects water-related data directly from the cabinet. We called the script at 10 second intervals and stored the output to disk for post processing. Note that this is a stop-gap solution which will be discontinued once this data is included in the SEDC stream.

IV. Workload Description

For this work, we developed an application work package to exercise the machine under a variety of conditions. Multiple iterations of single and multiple node runs of a combustion code, HPL, and HPCG were run as a group. This group was first run normally and then in turbo mode. These repeated cases we refer to as a *Series*. This *Series* was then run 3 times, once under each of the following conditions: no power capping (415 Watts), 50% node level power capping (322 Watts), and 0% node level power capping (230 Watts).

HPL is the MPI implementation of the high performance Linpack benchmark [10]. A highly regular dense LU factorization, HPL is computationally intensive. High Performance Conjugate Gradient (HPCG) [11] is a very different benchmark, by design. HPCG comprises operations such as sparse matrix-vector products that stress the memory subsystem and network communications, and its performance may be uninhibited by modest reductions in CPU resources or frequency. HPL and HPCG represent extremes in the spectrum of applications, from compute-bound to memory-bound, with orders-of-magnitude differences in Flops. In June 2014, Tianhe-2 reported the top numbers in both benchmarks, with 33.9 Pflops on HPL but only 0.58 Pflops on HPCG [12]. In addition to these benchmarks, we also included in our system evaluation an application code for direct numerical simulations of turbulent combustion. It uses an explicit Runga-Kutta method with mostly nearest neighbor communications.

V. Analysis and Results

In this work we combine information from log and numeric data sources from both facilities and our ART platforms to expose issues facing all large scale HPC host sites as we move to pre- and exascale platforms. In this section we first present the analysis methodologies used in this work. We then present both visual and analytic results of interest derived from data taken over a two day period while running the workload as described in Section IV. The results section first discusses power related understanding obtained in our testing thus far. We then present cooling related information. Finally we show traffic and congestion related network metrics and briefly discuss their utility in understanding performance variation as we move to larger scale systems.

A. Analysis Methodologies

Our ultimate goal is to understand sub-systems and their relationships and to characterize system behavior in order to more optimally use the machine and to diagnose issues.

In this work we consider several methodologies for the analysis of data. These are highlighted below, with specific application in the following sections. It is important to note that for the material discussed here, the complete understanding relies on the integration of data from a variety of sources and from a combination of analysis methods.

1) Log Analysis: Baler [13] is a log message processing tool that extracts patterns from message streams, where a pattern is deterministically extracted from each message by marking pre-defined known words (like words in the English dictionary) as static fields, and unknown words as variable fields represented by a Kleen star (*) in the pattern. Some patterns discovered by Baler relevant to this work are presented in Figure 5.

```
Example patterns:

280 * * - -  Node * interrupt *=*, *=*, *=* *[*]: * * *
    Processor Hot

283 * * - -  Node * power budget exceeded! Power=*,
    Limit=*, * Correction Time=*


Example messages corresponding to patterns 280
and 283 respectively:

bcsysd 2080 - -  Node 2 interrupt IREQ=0x20000,
    USRA=0x0, USRB=0x80 USRB[7]: C0_PROCHOT CPU 0
    Processor Hot

bcpmd 2140 - -  Node 2 power budget exceeded!
    Power=340, Limit=322, Max Correction Time=6
```

Figure 5. Example of log message patterns and their corresponding messages.

We use Baler in this work to process all system logs, browse for interesting patterns, count the occurrences of patterns in time-node space, and generate plot files for visualizing where and when events of interest occur.

Because of the deterministic nature of the patterns, we can easily compare messages and their occurrences at different times and even between both Trinitite and Mutrino. Further, Baler functions without requiring any input from the user about patterns, format, or content. This is particularly valuable for entirely new systems where the exact message text and location may be unknown.

In this case there were 1.8 million lines of log files, collected over a 29 hour interval, that Baler distilled into 251 unique patterns. This reduction enables a system administrator to easily search for key issues in a manageable list and then drill down on a time interval or node set for greater detail.

A visualization of the pattern occurrences from Figure 5 in node-time space is presented in Figure 9 in order to help analyze where and when particular events occur. This is described in greater detail in Section V-A. It is interesting to note that pattern 283 (power budget exceeded), was discovered in the controller directory logs and not in the power_management log, as might have been expected. (Note: Baler reserves the first 128 pattern IDs for internal use. Thus pattern numbering begins at 128).

2) Numeric and Visual Analysis: In addition to the log analysis, we use visualizations of integrated data and simple numerical analyses. We examine data time-series in relation to events in order to get an overview and understanding of system and variable behaviors and to discover and further refine our analyses of situations of interest. We examine data in the physical machine layout in order to detect physical system relationships. Finally we consider numerical analysis in order to determine abnormal behaviors.

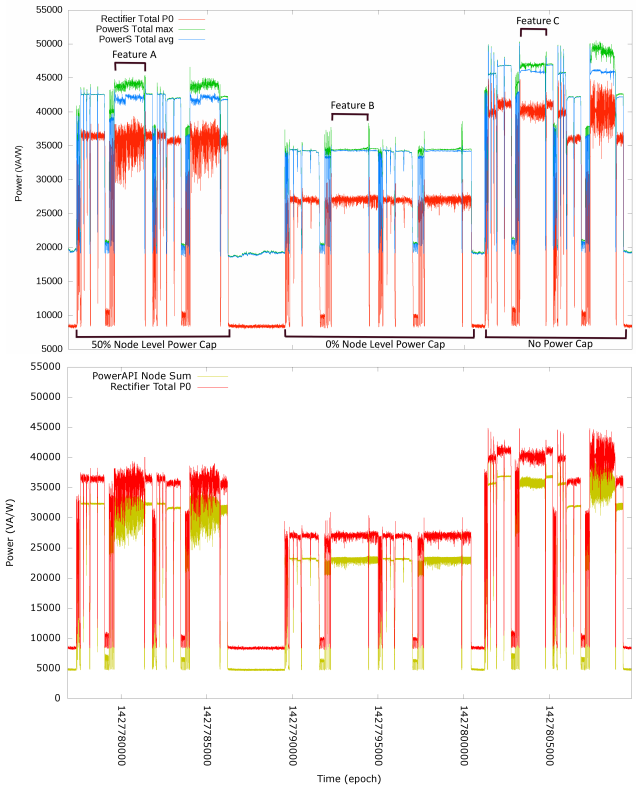


Figure 6. Power information from Facilities (green (max), blue (ave)), System measured at a cabinet level on the DC side of the rectifier (red), and Node Level data (gold).

B. Perspectives in Power Use: Facilities vs. Machine

In order to perform effective power management at a platform level it is necessary to understand the relationships between the platform's view of power draw and that of the physical plant (*facilities*). Thus we not only collected power and energy data from a variety of sources including in-band on the compute nodes at 10Hz, via SEDC at 1Hz, and energy from Cray's Resource Utilization Reporting (RUR) facility. But we also collected the *facilities* view (what really matters) using the methods described in Section III-E.

Figure 6 (top) shows a combination of three power views. PowerS_Total_avg (blue) is plotted as a time series of 5 second averages of the total for PowerS (this is the complex sum of PowerS over all three power phases and represents what the Utility company sees). PowerS_Total_Max (green) represents the maximum power draw over the past 5 second window. The 5 second data was obtained at the facilities level as described in Section III-E. The platform view of the average power draw over the previous 1 second window is represented by Rectifier Total PO (red). The discrepancy between facilities average power (blue) and peak power (green) over the time our applications (Section IV) were being run on Trinitite varies as the amount of fluctuation of power draw by the compute nodes (see Figure 7) i.e.,

the higher the peaks in the system data, the greater the discrepency between average and peak in the facilities data. As can be seen in the figure (top) the platform's view of power draw is about 20% lower than the data center infrastructure view. Some of this discrepency comes from loss in the rectifiers themselves. This power is not recorded in the SEDC data. This must of course be taken into account by the power management software. Additionally the power factor variation can skew this discrepancy. Under normal conditions (85 percent utilization) the power factor is satisfactory; however, if the utilization runs below these parameters, overall power utilization can be less efficient.

We call out Features A, B, and C in Figure 6 as time windows of interest over which we also plot 10Hz power data taken from the 100 compute nodes during a series of application runs. We chose these regions because they are wide enough and stable enough to be able to discern obvious differences in the behavioral characteristics of data center and machine based perspectives on power both within a region and between similar regions where power capping is the difference. Additionally the behaviors of the compute nodes with respect to the power caps for each region are clear (See Figures 7 and 8).

Figure 6 (bottom) shows compute node power data collected at 10 Hz and summed over all compute nodes for each 100ms collection period (gold) vs. the same Rectifier Total PO (red) plotted on the top trace. As expected the summed power draw as seen by the compute nodes is less than the total for the platform (by about 10%). Note that this system is only 2/3rds populated and this discrepancy will change on a fully populated rack.

In this set of application runs there was a node (nid00176) that did not change its power cap correctly and remained uncapped the whole time. Figure 8 shows a plot of the power usage of nid00176 versus another compute node plotted for comparison over region B. The failure to change caps was reported in the logs and would need to be taken into account by the resource manager in order to do appropriate power management on a large scale system where many such failures would be expected. While nid00176 was stuck in the no-cap state in this case, it seems equally probable that a node could get stuck in a lower power cap state. In such a case, if the resource manager naively handed the node out to a job that was running with no-cap the reduced performance of this node would adversely impact the whole job and could even cost more energy overall.

The plots in Figure 7 show an overlay of time-history plots of the 10Hz power data collected, using PowerAPI, over all compute nodes during time intervals labeled as Feature A, B, and C in Figure 6. These plots correspond to: 50 percent node level power cap (top) (Feature A in Figure 6 (top)), 0 percent node level power cap (middle) (Feature B in Figure 6 (top)), no power cap (bottom) (Feature C in Figure 6 (top)). The maximum power defined by the cap level is shown as a horizontal yellow line in each figure. The 10Hz data shows values that exceed the cap. Note that nid00176, which did not respond to the cap command is not included in the figures but is shown separately in Figure 8. Generally, data centers have to account for the fact that power capping is not an absolute.

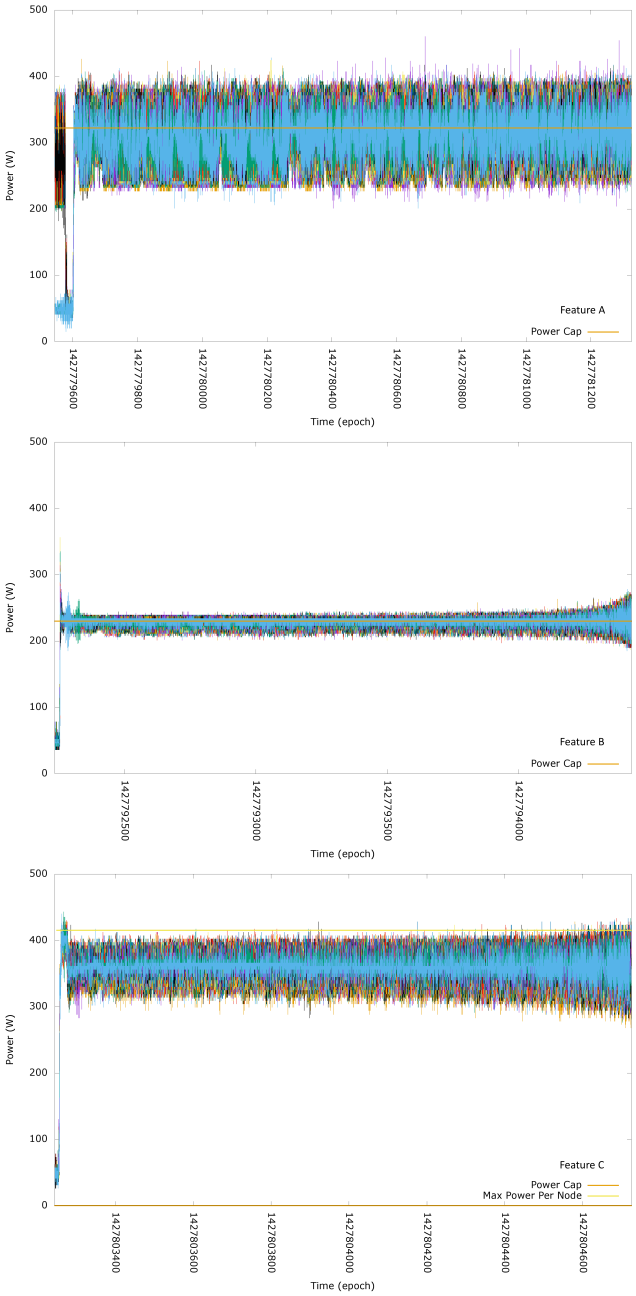


Figure 7. Plots of compute nodes' 10Hz power profiles for the cases of 50% node level (Feature A), 0% node level (Feature B), and no (Feature C) power cap from top to bottom respectively. Significant usage above the power cap is exhibited and has to be taken into consideration in power and performance decisions.

It is interesting to note that the noisiest of the three

features is A in which there is a 50% power cap and in which compute nodes regularly exceed the cap by up to 25%. This can also be seen in the Baler plots of Figure 9 which shows many more ”power budget exceeded” log message occurrences (red) than in the 0% cap region. In fact in region A the compute nodes regularly spike to the same levels seen in region C which has no cap though Figure 6 (top) clearly shows both the average and peak from the data center are higher (and less noisy).

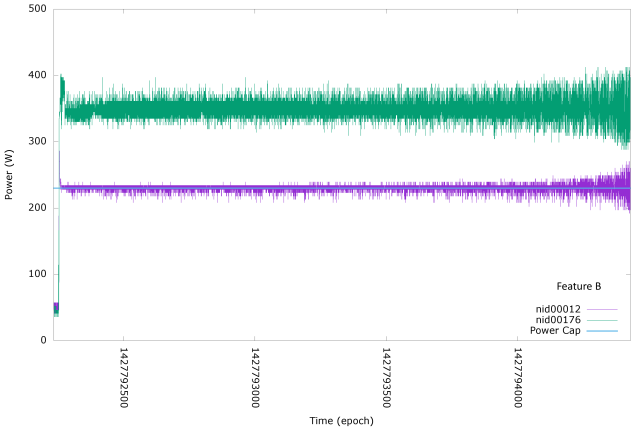


Figure 8. Non-capped behavior of nid00176 vs an arbitrary nid with 0% node level power cap (Feature B).

Figure 9 shows the time and location of occurrences of Baler [13] patterns representing ”power budget exceeded” (pattern 283) messages (red) and ”processor hot” (pattern 280) messages (green). The P283 messages are mostly seen during our application runs at a 50% power cap while the P280 messages were all during no-cap application runs. Note that though the P280 messages indicate a hot processor, there was no indication of corresponding temperature related throttling events. (Thermal distributions are considered in more detail in Section V-E).

C. Applications, Power, and Performance Perspectives

In this section we concentrate on the single-node runs, described in Section IV, in order to better understand the node to node performance variability that occurs when operating under a power cap. Due to part-to-part manufacturing variability, the power required to operate at a given performance level will be different from processor to processor, and hence node to node. Operating under a power cap should fix the maximum power used by a node, at least in theory, while allowing performance to vary. This is in contrast to setting a P-state, which results in a relatively fixed performance level across nodes, but with a variable power usage.

For each run we recorded the performance of the benchmark (i.e., HPL, HPCG) as reported in the benchmark's normal output, as well as the average power used by the

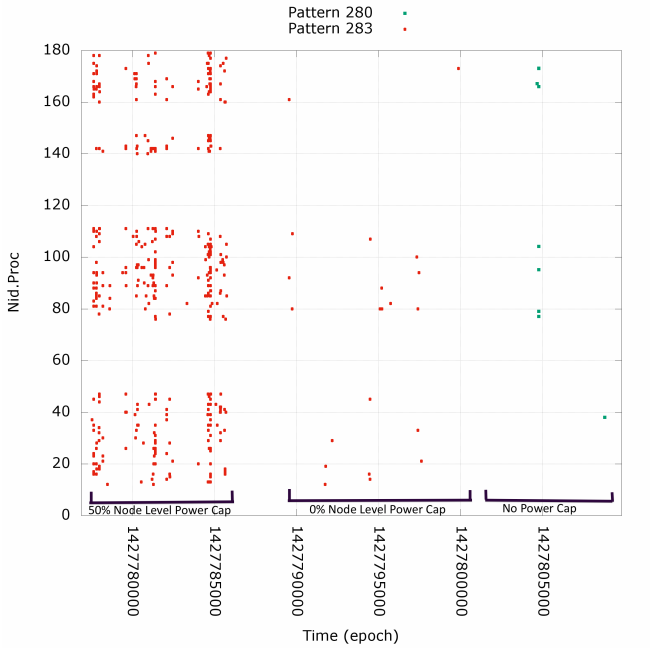
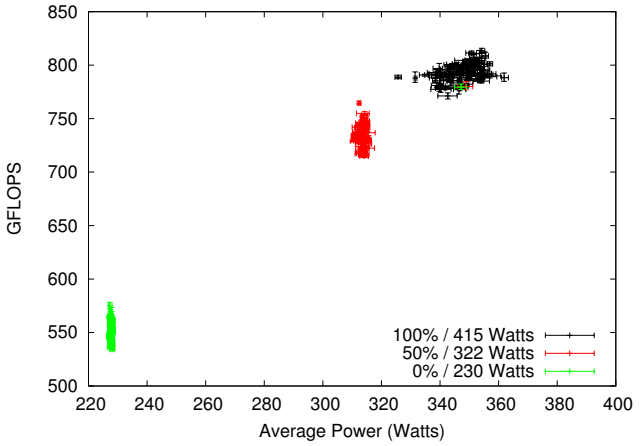


Figure 9. Baler error patterns relating to power (280 - Processor Hot, 283 - power budget exceeded, see Figure 5 for full patterns and example messages). Power budget is exceeded when the cap is set. Some processors are reported hot when no cap is applied.

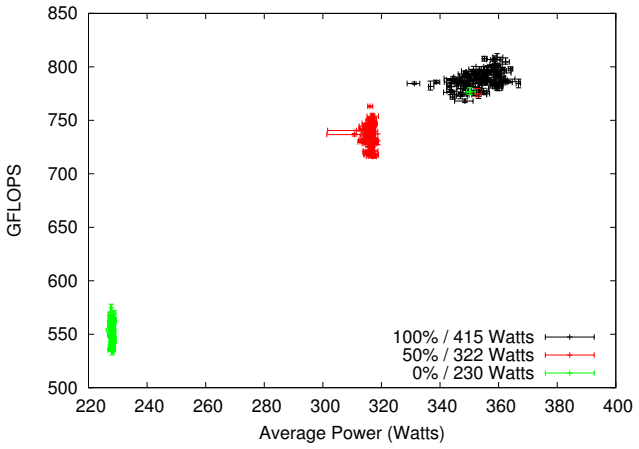
run over its entire execution, as reported by Cray's power measurement infrastructure. Each benchmark was run five times on each node for each of the three power cap configurations tested, 0%, 50% and 100%. Additionally, we tested with Intel's turbo boost feature on and off for each power cap configuration. For the Intel Xeon E5-2698 v3 (Haswell) processors used on Trinitite, the base non-turbo frequency is 2.3 GHz. Enabling turbo boost allows the processor's frequency to scale up to 3.6 GHz based on the number of cores being used and thermal headroom.

Results for the single node experiments are plotted in Figures 10 and 11, for HPL and HPCG respectively. In these plots each point represents the average value for the five trials and error bars (on both x and y axis) represent the minimum and maximum values recorded. HPL in the 100% power cap configuration (no power cap) results in a node-to-node performance spread of 767 to 812 GFLOPS and a power spread of 331 to 367 Watts. Since there is no cap in this configuration, each node is operating at the maximum speed it is able to, which depends on the energy efficiency of the processors in the nodes, environmental conditions, and other factors. In contrast, the 50% and 0% power cap levels result in a more vertical profile, indicating that the power cap is being hit. Each node uses as much power as it can, up to the power cap limit, and achieves a performance level based on the energy efficiency of the particular processors used in the node.

HPCG results, shown in Figure 11 show a much narrower

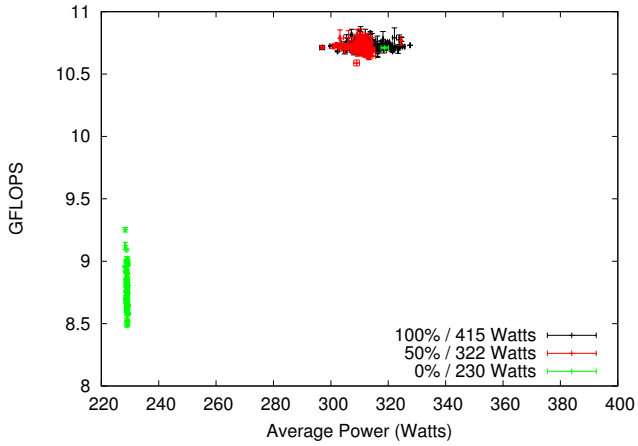


(a) No Turbo

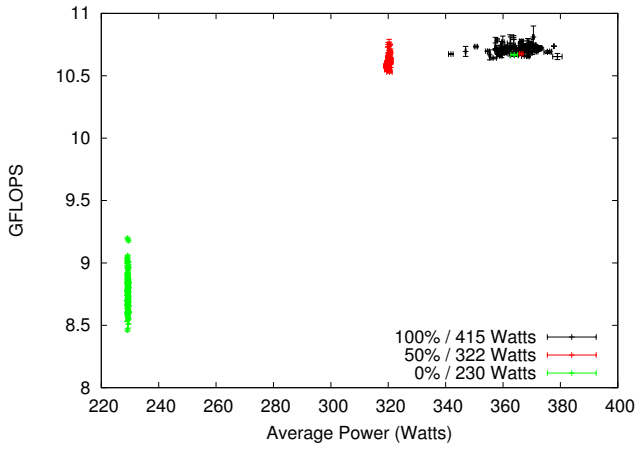


(b) Turbo On

Figure 10. Performance vs. power for HPL with different power caps and turbo off (top) and turbo on (bottom). nid00176 failed to implement the power cap.



(a) No Turbo



(b) Turbo On

Figure 11. Performance vs. power for HPCG with different power caps and turbo off (top) and turbo on (bottom). nid00176 failed to implement the power cap.

band of performance for the different power cap levels. The 100% and 50% configurations result in essentially the same performance levels, indicating that the power cap is not being reached. The 0% configuration leads to a sharply vertical profile due to the power cap being reached, and performance drops by approximately 17% compared to the other configurations. The HPCG 100% turbo on configuration is interesting because the processor is choosing to operate at a higher power level, even though it does not result in improved performance. This indicates that the processor's power management policy could be improved for HPCG, and likely other memory bandwidth bound codes as well (e.g., operate at the lowest frequency needed to saturate the memory subsystem). In both figures, nid00176 is the extreme outlier entity, coincident with the no cap group, since it did not respond to the cap command.

Figures 12 and 13 show histogram distributions of energy utilization of the nodes in the 0% and no-cap cases shown in Figure 11. The number of nodes is binned by percent of the average per-node energy for the application run (excluding nid00176) with the bin width being 1%. Both distributions look relatively normal but given the spread, a power aware resource manager should be able to take advantage of this when assigning nodes to applications. Knowledge of the run-time characteristics of an application could increase this advantage.

D. Machine Environmental Data and Facilities Interest

In this section we examine the cooling environmental data associated with the platform. This section delves into how platform information can be used for problem diagnosis and to gain facility infrastructure efficiencies. Data is presented from both Mutrino and Trinitite for comparison. For Trinitite, time ranges associated with applications during the work package in the non-turbo phase are marked. Times

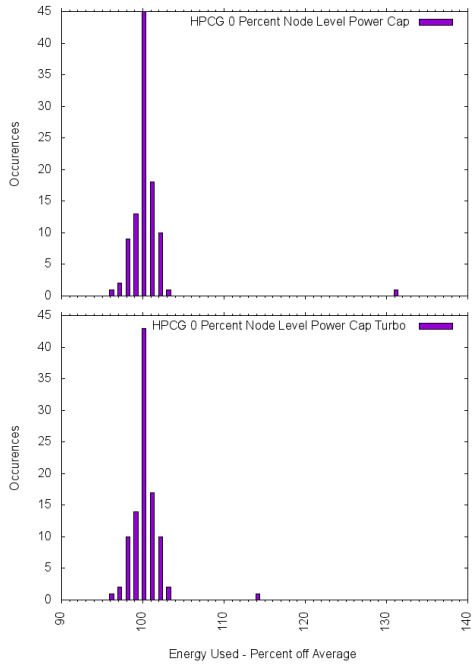


Figure 12. Distribution of number of nids vs. energy binned by percent of the average energy (average does not include nid00176, but the distribution does) for one set of the single nid runs HPCG with (bottom) and without turbo (top) and 0% node level power capping. Knowledge of variation of energy utilized for the same workload can be used in resource allocation decisions.

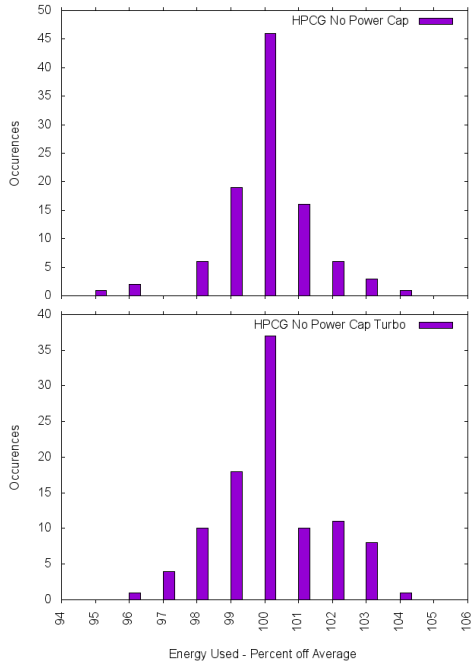


Figure 13. Distribution of number of nids vs. energy binned by percent of the average energy (average does not include nid00176, but the distribution does) for one set of the single nid runs HPCG with (bottom) and without turbo (top) and no Power capping.

in the figures are in HH:MM:SS format to facilitate easy assessment of the timescales of change.

Figure 14 (top) shows the valves opening in response to the jobs. The preconditioner valve position is calculated based on the inlet ambient temperature and dew point. The cabinet valve position maintains room neutral discharge air. The water line pressure varies during normal operation (middle). When the valve opens this results in lower pressure. This data can be used to determine pressure differential control and can also be used to calculate pump horsepower. Water outlet temp (bottom) is lower during the job because of the valve position. Facility personnel use these inlet and outlet temperatures to ensure the 75^oF water specification has not been exceeded. Monitoring the delta temperature helps to validate the coil is working effectively.

This type of information is not only a good indicator of data center conditions, but also could be used as input to data center building automation systems. In a tightly integrated system there is potential for automated control of pumps based on feedback from the platform environmental data. This type of automated control would allow for significant efficiency gains in the data center.

Figures 15 are from before and going into an HPL run on Mutrino that started at approximately 9:45. They are included to provide higher resolution insight into the behavior when a job is started and the temperature rises accordingly. In addition, they show some differences due to the water temperatures and differences in the blowers in the two systems.

In order to compensate for a higher altitude Trinitite was equipped with high volume fans, while Mutrino was not. Fan speed data can be used to proactively predict fan failures since the RPMs decrease as the fan reaches end of life. As can be seen in Figure 16 the Trinitite fans are set at constant 75 percent of max speed due to a software issue identified during facility acceptance testing. Mutrino fans are at the "normal" setting. Facilities fan data also shows that for both systems the fan power does not change.

E. Thermal Irregularities

Thermal issues are of interest for a number of reasons, including data center cooling, performance impact, and device degradation and aging. Issues in the components, machine, and the machine room environment may all be causes of such irregularities.

Numerical analysis of the SEDC data shows significant temperature variation across the CPU's of both systems when each was independently supposed to be running a similar workload across the nodes. Visual analysis of the numeric data in a physical layout (Figure 17) gives insight into this issue, as well as giving rise to further investigation.

The layout of compute nodes is generally as follows. On a blade, compute nodes are ordered $\{2, 1, 3, 0\}$ front to back. There are 2 CPU's per compute node. with CPU's 0/1

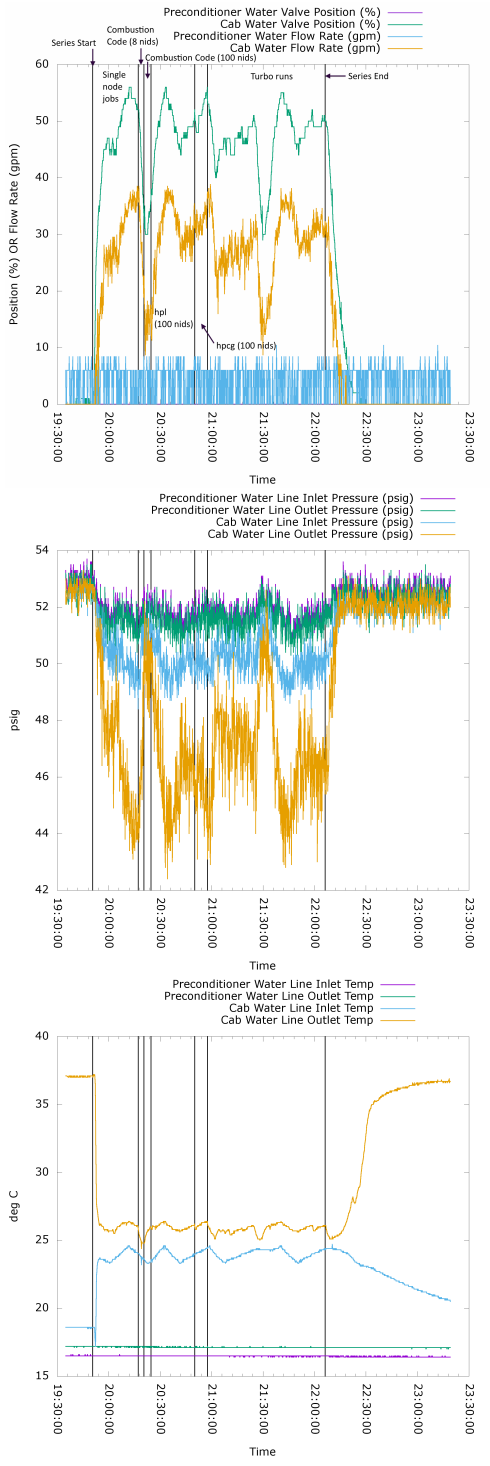


Figure 14. Trinitite machine data over the Series: Flow Rates and Valves (top) (Preconditioner water valve position is always 0.), Water Line Pressures (middle), Water Outlet Temp (bottom). We seek understanding of the response of the machine to workloads to enable automated control of the facilities as a whole.

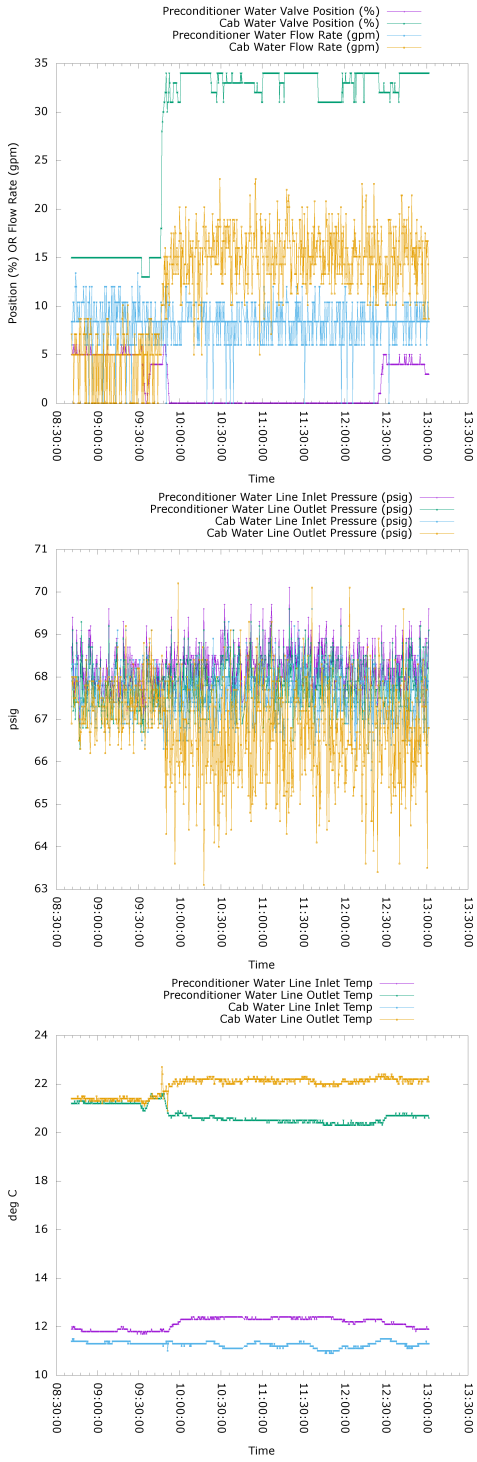


Figure 15. Mutrino data for comparison, going into an HPL run that started at approximately 9:45. Flow Rates and Valves (top), Water Line Pressures (middle), Water Outlet Temp (bottom).

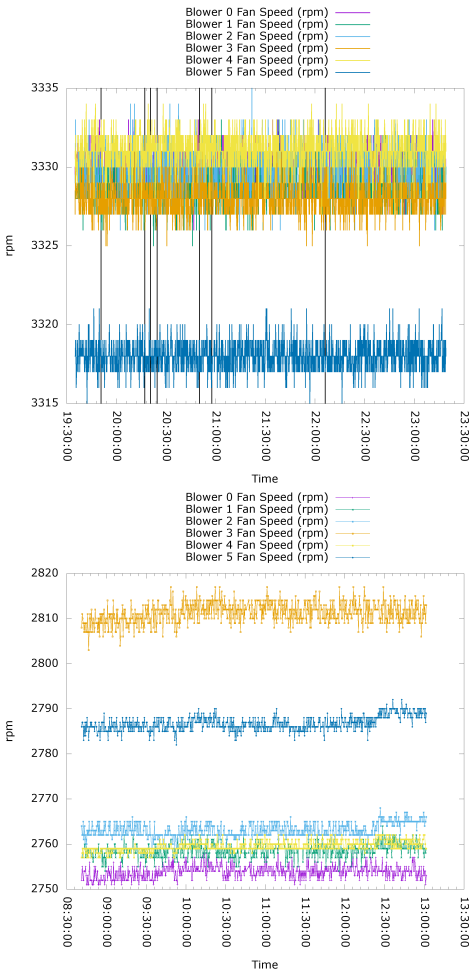


Figure 16. Blower Speed: Trinitite data over the Series (top). Mutrino going into an HPL run (bottom).

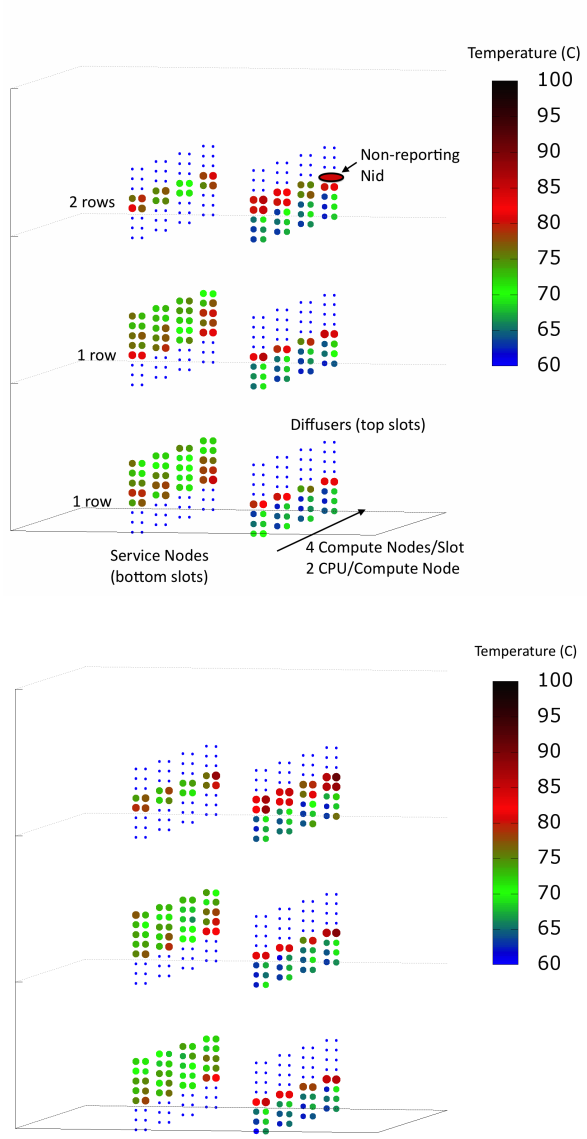


Figure 17. Thermal distributions on Trinitite (top) and Mutrino (bottom) shown in the layout of the rack (not to scale). Air flows left to right. There is significant temperature variation across the system and across CPUs on a slot. Temperatures are markedly higher when the left and right slots are both populated (overlap rows labeled in top figure). c0-0c2s12n0/nid00176 is of interest in both systems.

alternating left/right with each node. There are two service nodes per blade. Within the rack, chassis are vertically stacked. Two slots can be populated left and right in a chassis.

Figure 17 shows the layout for the two machines (Trinitite (top), Mutrino (bottom)). Slots with Service Nodes are located on the bottom left slots of each chassis; diffuser slots are located at the tops of the chassis; either are indicated by the small blue dots (color not indicative of temperature for non-compute nodes).

Maximum CPU temperatures for each compute node over the workload for each case are shown, with the exception of Trinitite's c0-0c2s12n0, as discussed below. The workloads were not the same for each machine. For Trinitite, the workload was the entire set of runs discussed in this work (3 *Series*). For Mutrino, the workload was the entire HPL run (approx 3 hours) pertaining to the Mutrino figures in Section V-D. While it is not expected for the values to be comparable, certain similarities occur in both.

There is a significant overall variation in CPU temperature (25-30^oC). These large differences could lead to differences in component aging, performance, and failure rates. In addition, there can be a greater than 10^oC temperature variation across CPUs in the same blade. In general, the hotter nodes are seen to be those for which the left and right slots are both populated. In addition, c0-0c2s12n0, which is nid00176, has issues in *both* systems: in Mutrino this node has been exhibiting temperature related throttling; in

Trinitite the SEDC data included error codes for all attempts at collecting temperature related data for this run and in the log data this nid was the only nid reporting an error when attempting to apply the power capping profiles. After further discussion with Cray [14], we believe these Trinitite issues may be related as the same communications channel is used to issue the power capping command and to obtain the SEDC data. As of this writing, the occurrence of this communications failure remains unresolved. We do not have enough information to determine if the situation might be thermally related.

As a result we seek further understanding of the common positional dependence of the problem nid, of the overall expectations of temperature within the partially populated racks, and of how we can expect the results to extrapolate to fully populated systems.

F. Network

While the workload did not target investigation of network performance and Aries environmental data profiling, the Aries does consume energy and understanding when contention for network resources (congestion) is affecting performance can aid in root cause analysis of performance variability and help in optimization of job placement that minimizes congestion. We briefly present a first look at this data in the form of link bandwidth used and associated stalls (a measure of congestion).

Figure 18 plots network traffic (top) and stalls (middle) data for each of the 40 outward facing Aries router tiles. This data was gathered from the gpcdr interface via LDMS (Section III-D) during the course of the same application runs previously presented. As expected there are no traffic or stall values during the single node jobs and highest values occur during the 100 node combustion code run. (Only the times in the first set of runs in the *Series* (non-turbo) are marked.) The bottom figure shows related SEDC environmental data on the Aries associated with the entire blade associated with that node. Note all nodes of a blade share an Aries router and the data shown here, though collected by one node, is for all traffic and stalls associated its Aries router. For this workload, there is only a slight, but noticeable, effect on the current (bottom) during the runs. Future work involves consideration of more communication intensive workloads.

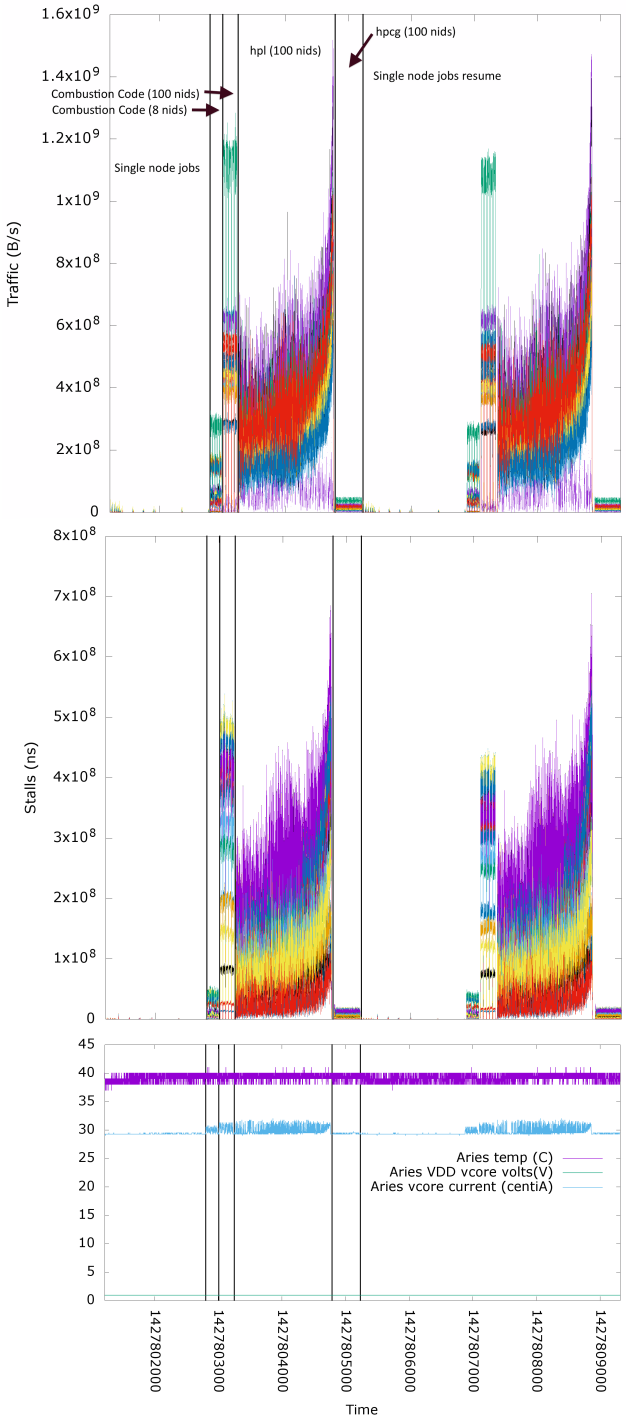


Figure 18. Integrating Network traffic and environmental data. Traffic (top), stalls (middle) collected on an arbitrary nid during the no capping case. Aries SEDC values for the slot of that same nid (bottom).

VI. Conclusions and Future Work

Integration of data from a variety of sources is necessary for system understanding, improving system performance, and problem diagnosis. This becomes increasingly necessary as we continue to push the boundaries of the data center infrastructure supporting HPC systems.

In this paper we have considered information integration and analysis functionalities currently deployed and under development on the ACES ART systems for Trinity. Data

sources include facilities, machine, and node level data and include both numeric and log data types. Our monitoring architecture is intended to support run-time analysis and decision-making based on the data. We presented actual cases of analysis of the integrated data relating to power, cooling, and thermal issues and areas of interest. In particular, as we have targeted enabling more advanced facilities operation, our integration and analysis of this data has been key in identifying the need for a more clear understanding of the differences between machine and data center power reporting.

Future work includes the deployment of these monitoring capabilities on Trinity and further analysis and understanding of system behaviors and abnormalities particularly as they relate to power, thermal, and networking issues. We seek to capture metrics from the platform which can then drive infrastructure efficiencies through automating dynamic facility control. We seek to enable capabilties such as power capping and load shedding from the platform to respond to facility power and cooling constraints. We plan to characterize behaviors in order to predict optimal curve ratios from pump and tower perspectives. We will analyze ratios of power usage of components within compute hosts (power supplies, DIMMS, and CPUs) in order to drive more specifications for more efficient architectures in future procurements. Ultimately, our long term goal is to use our enhanced understanding to enable advanced operations of the site facilities in concert with the site's machines operations.

ACKNOWLEDGMENTS

The authors would like to thank Jason Repik (Cray) for configuration, advice, and diagnostics; Paul Casella (Cray) for information and fixes to Cray's *gpcdr* module; Joshi Fullop (NCSA) and Victor Kuhns (Cray) for useful discussions on the forwarding of SEDC and log data; Adam DeConinck (LANL), Kathleen Kelly (LANL), and Jim Williams (LANL) who administer the platforms and facilitated the runs used in this work. and Alynna Montoya-Wuiff (LANL) and Eloy Romero (LANL) for access to facilities data.

REFERENCES

[1] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," *SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 77–88, Jun. 2008. [Online]. Available: http://doi.acm.org/10.1145/1394608.1382129

[2] "Using and Configuring System Environment Data Collections (SEDC) Cray Doc S-2491-7001," 2012. [Online]. Available: http://docs.cray.com/books/S-2491-7001/S-2491-7001.pdf

[3] P. Falde, private communication.

[4] J. Laros, D. DeBonis, R. Grant, S. Kelly, M. Levenhagen, S. Olivier, and K. Pedretti, "High Performance Computing - Power Application Programming INterface Specification, Version 1.0," Sandia National Laboratories, Albuquerque, New Mexico 87185 and Livermore, California 94550, Technical report SAND2014-17061, 2014.

[5] "Monitoring and Managing Power Consumption on the Cray XC System Cray Doc S-0043-7202," 2014. [Online]. Available: http://docs.cray.com/books/S-0043-7202/S-0043-7202.pdf

[6] 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. IEEE/ACM International Conference for High Performance Storage, Networking, and Analysis (SC14)*, 2014.

[7] P. Cassella, private communication.

[8] "Trane." [Online]. Available: http://trane.com

[9] C. McMurtrie, L. Gilly, and T. Belotti, "Cray Hybrid XC30 Installation - Facilities Level Overview," in *Cray User's Group*, 2014.

[10] "HPL." [Online]. Available: http://www.netlib.org/benchmark/hpl/

[11] "HPCG." [Online]. Available: http://www.hpcg-benchmark.org

[12] "HPCG Performance." [Online]. Available: https://software.sandia.gov/hpcg/2014-06-hpcg-list.pdf

[13] N. Taerat, J. Brandt, A. Gentile, M. Wong, and C. Leangsuksun, "Baler: deterministic, lossless log message clustering tool," *Computer Science - Research and Development*, vol. 26, no. 3-4, pp. 285–295, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00450-011-0155-3

[14] S. Martin and D. Rush, private communication.