Cray XC30 Power Monitoring and Management

Steven J. Martin
Cray Inc.
Chippewa Falls, WI USA
stevem@cray.com

Matthew Kappel
Cray Inc.
St. Paul, MN USA
mkappel@cray.com

***Abstract*—Cray customers are increasingly demanding better performance per watt and finer grained control of total power consumption of their data centers. Customers are requesting features that allow them to optimize application performance per watt and to conduct research in support of future system and application power efficiency. New system procurements are growingly constrained by site power and cooling limitations, the cost of power and cooling, or both. This paper describes features developed in support of system power monitoring and management for the Cray XC30 product line, expected use cases, and potential future features and functions.**

***Keywords*-Power monitoring; power capping; RAPL; energy efficiency; power measurement; Cray XC30**

I. Introduction

Cray customers are increasingly demanding better performance per watt and finer grained control of total power consumption in their data centers. Some features toward that end are to enable users and data centers to optimize application and library performance per watt, bill users for job power consumption, and to conduct research in support of system energy efficiency. At the same time new system procurements are increasingly constrained by total site power and cooling limitations either in cost or infrastructure. It is now common for new system bids to be driven by total cost of ownership (TCO) where the total energy cost for the life of the system can have a dramatic effect on the amount of hardware that can be proposed to meet customer requirements. Use of overly conservative power estimates when developing system proposals can result in reduced overall system capabilities and over-provisioning of site-level infrastructure. As a result, it benefits everyone in the HPC community to work toward informed and reasoned energy efficiency.

Toward that goal, Cray has made substantial investments in its software and hardware capabilities for energy efficiency all the while continuing to develop best-in-class extreme scale systems. In the near future, the development and runtime management of systems will necessitate delivering exascale performance within a targeted 20 MW power envelope [1] [2]. It is clear to Cray that both software and hardware will need to play a part in that engineering.

Accurate and timely power monitoring data is critical for evaluation and development of hardware and software features in general, as well as those used to control system energy consumption [3]. Power monitoring of Cray systems will be extended over time to enhance current and introduce new capabilities. Access to finer grained power monitoring and control is needed to support research and development in advanced power management. Cray is committed to understanding changing industry and customer requirements and is driving to meet these requirements as future hardware and software designs are developed [4] [5].

In this paper we will first outline the power monitoring and management features now available on current Cray XC30 systems and give some details as to how they are implemented. Section II describes how the current features can be used to implement some common, while section III discusses intended use cases of interest to customers. Following that, section IV previews new features currently being developed and some of the features planned for implementation in the 2014 - 2015 time frame.

II. Current XC30 power monitoring and management functionality

The initial release of base software support for XC30 power monitoring and management functionality shipped in June of 2013 as part of the System Management Workstation (SMW) 7.0.UP03 and Cray Linux Environment (CLE) 5.0.UP03 releases. Those releases included support for Cray Marble blades with Intel Sandy Bridge processors, however that support was limited in that data collection into the Power Management Database (PMDB) was disabled by default. Customers interested in experimenting with the new power monitoring features in the first release needed to manually enable data collection.

Subsequent releases have added some additional features and refinements. The first major update in September 2013 as SMW 7.1.UP00 and CLE 5.0.UP00 enabled data collection into PMDB by default, These releases included Resource Utilization Reporting (RUR) support of application energy reporting with its energy plugin, and power management and monitoring support for Cray Graphite Blades with NVIDIA K20 GPUs as accelerators.

The most recent software releases: Cray perftools/6.1.3 (December 2013), SMW 7.1.UP01 (December 2013), and

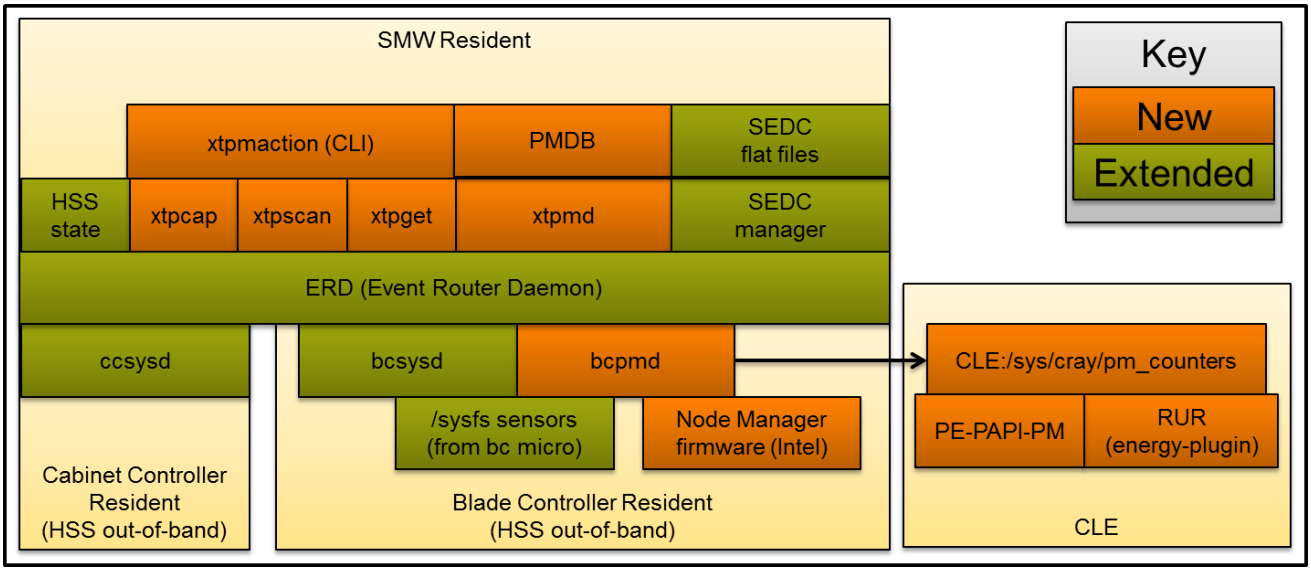

Figure 1. Cray power monitoring and management software stack

7.2.UP00 (March 2014) provided additional power monitoring and management functionality: support for Intel Xeon Phi coprocessors, NVIDIA Tesla K40 GPUs, enhanced RUR user level reporting capabilities, and Cray Programming Environments (PE) support for Intel Running Average Power Limiting (RAPL) and custom Cray pm_counters.

The high level power monitoring and management software stack represented in Figure 1 calls attention to some of the major functional blocks of current Cray power monitoring and management stack.

A. XC30 System Environment Data Collection

The Cray software distribution includes System Environment Data Collection (SEDC) software which supports collection of system environmental data including cabinet-, blade-, and node-level sensors for temperature, current, and voltage. The SEDC monitoring infrastructure has been part of the Cray HSS software stack for several product generations and has been been enhanced in further support of power monitoring goals. SEDC updates for XC30 include the addition of all available current sensors and many new sensors provided by the Intel Xeon chipset. Section IV-B discusses future plans to convert SEDC from its current use of flat files for data storage to using the PMDB as its backing store.

B. High speed out-of-band power/energy monitoring

A new high speed, out-of-band power/energy data collection capability has been developed for the Cray XC30. This new high speed capability enables collection of cabinet-, blade-, and node-level power, energy, and related data. This software has been designed to be extensible as hardware with finer-grained monitoring capabilities becomes available.

At the blade-level, a new blade controller daemon *bcpmd* collects node and Aries Network Card (ANC) point-in-time power at 10 Hz. The 10 Hz point-in-time power data is then used to maintain a 64-bit accumulated energy counter for each of the four nodes and the ANC on each blade. This point-in-time power and accumulated energy data is sent from the blades up to the SMW at a default effective rate of 1 Hz. Note that power and energy data collected at the blade-level is buffered before it is sent to the SMW. This data buffering prior to transport to the SMW, using the Cray event router, is done in order to optimize performance and minimize small packet processing overhead over the Hardware Supervisory System (HSS) network. The existing cabinet controller daemon *ccsysd* has been updated to collect cabinet-level power (for both the rectifiers and blowers, where applicable), accumulated energy, current, and voltage data which is sent to the SMW at a static rate of 1 Hz.

C. Power Management Database

On the SMW, all power and energy related data is collected by a new daemon *xtpmd*. This daemon buffers and inserts this data into the new PostgreSQL-based [6] PMDB. The xtpmd daemon also listens for application start, stop, suspend, and resume events from ALPS. Upon receiving these events, xtpmd updates PMDB tables tracking: job start/end time, job-id, application-id, user-id, and application node allocation data. Power and energy data at the component-level for cabinets and blades is stored in the *pmdb.cc_data and pmdb.bc_data* tables, respectively. Both of these tables are partitioned to allow for manageable indexing and their total size is limited by per-table parameters: maximum number of table partitions and the maximum number of rows per table partition. The pmdb.cc_data and

pmdb.bc_data tables maintain a configurable maximum size by table partition rotation: dropping the oldest partition and adding a new table partition for new data.

A 'hooks' interface allows for specifying scripts to execute when a partition is about to be dropped. Along with the data table sizing parameters, this hook interface are intended to allow for site-level customization and can be managed via the *xtpmdconfig* utility on the SMW. A potential use for this interface might be to periodically backup data to an external disk. The default hooks that ship with the software release remove entries from the job-related tables (`pmdb.job_timing` and `pmdb.job_info`) when raw node-level data for given jobs is dropped. Note that without the default hook functionality the pmdb.job_timing and pmdb.job_info tables would grow un-bounded.

Figure 2 shows an overview of the user-facing PMDB schema. In addition to the aforementioned job and component-level data tables, PMDB provides metadata tables to describe that data and aid in development of custom SQL queries:

- *pmdb.sensor_info*: Lists the id, name, and units for each sensor referenced in the pmdb.cc_data and pmdb.bc_data tables,
- *pmdb.nodes*: Faciliates lookups between NID values stored in the job tables and component name values which are used to query the component-level data tables, and
- *pmdb.sensor_spec*: Details hardware-specific sensor specification information.

For more details, refer to Chapter 3 of [7]. Readers interested in accessing the PMDB should pay attention to the SQL helper functions `source2cname(`*`source`*`)` and `cname2source(`*`'cname'`*`)` which are also described in [7] and their use is demonstrated in the example scripts that ship with the SMW release *smw:/opt/cray/hss/default/pm/script_examples*.

D. Accelerator (GPU/MIC) power/energy data collection

In addition to the node-level data listed above for two socket nodes, point-in-time accelerator power and accelerator accumulated energy data is collected for XC30 blade types that include accelerators. Low level data collection of accelerator data is also accomplished at 10 Hz, leveraging the same infrastructure developed for the non-accelerated blades. The hardware components and software that collect accelerator data is the same for the blade types that support the NVIDIA GPU (Cray Graphite) and Intel Xeon-Phi (Cray Granite). As with the node-level energy counters, the accumulated accelerator energy counter is maintained by HSS software in a 64-bit register that is not expected to rollover and will only reset on a blade-controller reboot or power cycle.

```
/sys/cray/pm_counters/accel_energy:24675886 J
/sys/cray/pm_counters/accel_power:22 W
/sys/cray/pm_counters/accel_power_cap:0 W
/sys/cray/pm_counters/energy:71224823 J
/sys/cray/pm_counters/freshness:4516770
/sys/cray/pm_counters/generation:9
/sys/cray/pm_counters/power:62 W
/sys/cray/pm_counters/power_cap:425 W
/sys/cray/pm_counters/startup:1396011015159068
/sys/cray/pm_counters/version:1
```

Figure 3. Snapshot of /sys/cray/pm_counters on a *(Graphite)* node

E. In-band access to power/energy monitoring

Blade level HSS software publishes node (and accelerator) power, energy, and related data collected out-of-band in a way that allows (on-demand) in-band access from CLE via special sysfs files in /sys/cray/pm_counters. This in-band access path will not cause system overhead or application jitter when the data is not being accessed. The /sys/cray/pm_counters (Figure 3) include all of the 10 Hz data indicated above, as well as some other useful information including power cap settings that may be in place. The /sys/cray/pm_counters files are read-only and available to (unprivileged) users and debug, performance, and system accounting utilities.

Other 'meta-counters' available in /sys/cray/pm_counters are worth brief mention. The *'freshness'* counter is incremented by the lowest level firmware on each pass through the loop reading the hardware sensors. This can be used to test data quality. The freshness counter should incrementing at approximately 10 Hz. If the freshness counter stops incrementing, all of the other counters are invalid. The *'generation'* counter is incremented any time a power capping change is made. By reading the generation counter before and after a code is run, a change in the power cap can be detected even if the initial and final *'power_cap'* counter readings match. The use of the generation counter this way should be considered a hint for debugging. Finally, the *'startup'* counter is included in order to allow for the detection of a blade controller reset.

Given all of this, the following rules should be considered when comparing two snapshots of files in the */sys/cray/pm_counters* directory on any given node:

- The *'startup'* counters must match.
- The *'freshness'* counter incremented at a rate of approximately 10 Hz.
- Accumulated *'energy'* (and *'accel_energy'*, if applicable) counters are monotonically increasing.
- Node-level *'power'* counters must never be zero.

Although it is not strictly necessary, some software will read all of the counters twice and only capture a snapshot if the

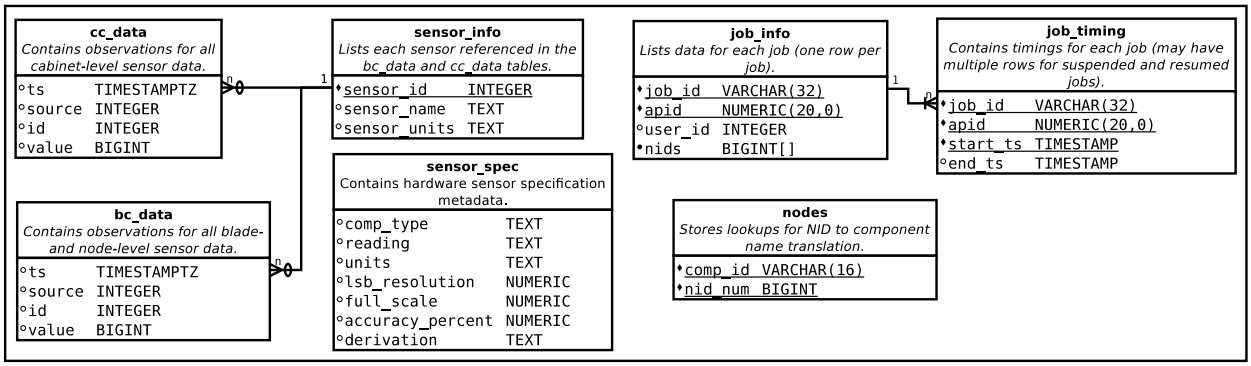

Figure 2. PMDB user-facing schema

'freshness' counter is the same in both. Most use cases are not impacted if some counters are 100 ms older than the others in a snapshot.

The Cray RUR software provides an energy plugin that uses the /sys/cray/pm_counters to report application energy usage. The RUR energy plugin calculates total application energy by reading node accumulated energy for all nodes assigned to an application, both at startup and completion, subtracting to get node-level energy, and finally summing energy over all nodes in a job. With several standard output options and the ability to be extended by the customer, RUR can be configured to get application energy usage information into the hands of system administrators, third party workload management logs, and directly to users. The latest RUR release in (CLE 5.2.UP01) also adds a user level opt-in for reporting data directly to user selected destinations. With this opt-in configuration option it should be much easier to enable user-level reporting at customer sites. Note that RUR is installed with CLE, but by default it is not enabled. RUR configuration information is available in the Cray publication S-2393 - *Managing System Software for the Cray Linux Environment* [8].

Cray Programming Environments software has integrated support for the Intel RAPL counters and /sys/cray/pm_counters in the perftools/6.1.3 release (December 2013). This support in PAPI [9] and CrayPat tools requires CLE 5.1.UP00 or newer to be installed on the system. For information on how to use this new Cray PE functionality on a live system try: `module load perftools/6.1.3; man rapl` [10].

F. System level power capping

System level power capping on XC30 systems is implemented in a layered approach using a combination of software, firmware, and processor hardware support. The XC30 system administrator interested in enabling, configuration, or modifying power capping settings will do so using the program *xtpmaction* running on the SMW [11].

The xtpmaction program is the single command line interface (CLI) program that is intended to support all out-of-band power monitoring and power capping configuration settings on XC30 systems. Power capping in xtpmaction is supported by creating, applying, and updating power capping 'profiles'. Power capping profiles allow the administrator to define worst case power limits for each blade type in the system. There may be multiple power profiles defined for a system, but only one profile can be enabled at any time (per partition or system). All blades supported in XC30 systems support a 'node' power capping control. Blades with accelerators (i.e., Cray Granite and Graphite blades) have an additional 'accel' power capping control. The power management configuration 'profile' uses key=*value* pairs where the 'node' and optionally 'accel' controls are keys, and the power cap value is expressed in watts. The node=*watts* key-value pair is used at the node level to enable node-level power cap enforcement via Intel's Node Manager firmware support for RAPL [12] [13].

On Cray nodes with accelerators, the optional [accel=*watts*] key-value pair can be used to limit that maximum GPU or MIC power consumption. Note that the 'accel' control is a subset of the 'node' control, and as such, is intended to allow for changing the balance of max-power between the Xeon host processor and the accelerator. The default for administrator-managed power capping is to only use the 'node' control and allow the accelerator to use as much power as it wants while staying within thermal limitations.

Initial support for system power capping on XC30 shipped with the SMW 7.0.UP03 software release. Support for new blade types including accelerators have been in subsequent releases, and Cray intends to continue to support power capping functionality on all future XC30 blades.

G. P-state at job launch

Cray XC30 systems have support for running jobs with a fixed p-state frequency. This functionality is provided is order to allow customers to save energy when they know that running with the Cray-default Linux 'performance' governor

is sub-optimal, desire to trade job-level power consumption for increased runtime, or conduct research. Research in the use of p-state by Laros et al. shows effective use of p-state to save energy on HPC applications [3] [14]. Cray's support for running jobs or applications with the Linux 'userspace' governor and a fixed frequency can by accessed using the `aprun --p-state=KHz` option, or via workload manager use of the Batch Application Scheduling Interface Library (BASIL) to request a fixed p-state frequency at job reservation time. Usage details are described in section 4.2 of [7].

III. Use cases for current XC30 power monitoring and management features

In this section, we will introduce some of the use cases envisioned and supported by the current power monitoring and management features of the Cray XC30 system. First, we will touch on some higher level topics, and then some simple examples will be presented, but not in great detail. Three other papers planned for CUG 2014 are expected to dive deeper into the use of some of the XC30 power monitoring and management capabilities from the system operations and applications development and benchmarking perspectives [15] [16] [17].

A. Access to realtime system power consumption data

First, we will describe the use of the *xtpget* command line tool. The use of xtpget is intended to be very simple, and not require the user to have any programming or database experience. The output of xtpget shows system power (current, rolling average, and peak), and accumulated system energy for a user specified count (`-c|--count`) number of samples. The user can request continuous output with a count of zero (`--count=0`). The default delay (`-d|--delay`) and window (`-w|--window`) settings are 1 and 10 seconds, respectively.

The output of `xtpget -c 15 -d 5 -w 20` is shown in Figure 4. The example output shows that each line of output starts with a timestamp followed by current power, average power, peak power, and accumulated energy for the system, with the average and peak power values for the requested 20 second window.

B. Access to system-level PMDB data from the SMW

For access to system-level power and energy data at a finer-grained level, users with SMW access can query the PMDB. Because of the flexibility inherent in collecting finer-grain data, a number of statistics can be calculated include system power, system accumulated energy, per-cabinet power statistics, per-cabinet cooling power statistics, etc.

For example, an aggregate of the power for all cabinets in the system at a point in time is effectively the total system power at that point in time. While cabinet power and energy values are collected and stored at a fixed 1 Hz cadence, the timestamps for all cabinets will not be at the same exact time to the millisecond, so it is necessary to use 'data binning' to adjust the values over the entire system to the same second and then perform the summation. Additionally, if a cabinet includes a blower, then both rectifier and blower power will need to be summed to arrive at the total power for that cabinet. Using this method, it would be possible to build a per-second time series of historical system power data that could be used in capacity planning.

In addition to the basic text-based queries mentioned above, some basic bash programs have been developed for in-house use that support the generation of plots visually conveying system and cabinet level power and energy data. The two plots in Figure 5 show Total System Power, the sums all of the compute cabinet and blower cabinet data available for the user's requested time interval (left), and Cabinet Power, only the compute cabinet power with each cabinet plotted individually (right). These plots help to illustrate the utility of looking at the data from more than one perspective. Having power data in an SQL database opens up opportunities for site-specific use cases. The system profiled in the plots shown in Figures 5 has multiple blade types, runs a mix of dedicated and shared batch time slots, and is likely not typical of a production data center.

C. Access to application-level PMDB data from the SMW

Personnel with authorized access to the SMW can access detailed power, energy, and application allocation data from the PMDB and log files on the SMW. In the following examples, we collect data for an application which was run two times, apruns corresponding to ALPS apids 1348984 and 1348989. The example script *cray_pmdb_report_energy_single_job.sh* was used to generate the text captured in two sets of text output captured in Figure 6. A slightly more complicated script was used to generate the gnuplot output captured in Figure 7. The script used generate the gnuplot leverages the concepts demonstrated in the cray_pmdb_report_energy_single_job.sh example script but also extracts point-in-time power information for each node in the application and feeds that data into gnuplot. In this example only the node (i.e., nid) assignments changed between the two runs, but you can see how this may be very useful in profiling the effects of many different types of application parameters.

D. System power capping

Perhaps the best use case for power capping on a Cray system is the use of a power cap that is set up to prevent unexpected power spikes above and beyond that of the normal workload of the target system. The first step in setting up this use case is to profile the system for some amount of time and get an understanding of the system- and node-level power consumption typical typical of a site.

```
crayadm@smw:~> xtpget -c 15 -d 5 -w 20
2014-02-31 16:50:49.073026  - Current Power 31124.00 (W) Average Power 30771.40 (W) Peak Power 31124.00 (W) Accum Energy 7845268553 (J)
2014-02-31 16:50:54.078609  - Current Power 30688.00 (W) Average Power 30747.00 (W) Peak Power 31124.00 (W) Accum Energy 7845422215 (J)
2014-02-31 16:50:59.084072  - Current Power 30052.00 (W) Average Power 30651.60 (W) Peak Power 31124.00 (W) Accum Energy 7845574067 (J)
2014-02-31 16:51:04.089529  - Current Power 30426.00 (W) Average Power 30665.40 (W) Peak Power 31124.00 (W) Accum Energy 7845726862 (J)
2014-02-31 16:51:09.094993  - Current Power 30862.00 (W) Average Power 30574.40 (W) Peak Power 30943.00 (W) Accum Energy 7845879847 (J)
2014-02-31 16:51:14.100450  - Current Power 30542.00 (W) Average Power 30530.60 (W) Peak Power 30862.00 (W) Accum Energy 7846032402 (J)
2014-02-31 16:51:19.105841  - Current Power 30367.00 (W) Average Power 30605.80 (W) Peak Power 31122.00 (W) Accum Energy 7846186033 (J)
2014-02-31 16:51:24.111266  - Current Power 30884.00 (W) Average Power 30657.10 (W) Peak Power 31122.00 (W) Accum Energy 7846340208 (J)
2014-02-31 16:51:29.116686  - Current Power 30185.00 (W) Average Power 30603.80 (W) Peak Power 31122.00 (W) Accum Energy 7846491626 (J)
2014-02-31 16:51:34.122141  - Current Power 30194.00 (W) Average Power 30633.20 (W) Peak Power 31122.00 (W) Accum Energy 7846644499 (J)
2014-02-31 16:51:39.127620  - Current Power 30177.00 (W) Average Power 30526.90 (W) Peak Power 31024.00 (W) Accum Energy 7846796235 (J)
2014-02-31 16:51:44.133053  - Current Power 30064.00 (W) Average Power 30450.60 (W) Peak Power 31024.00 (W) Accum Energy 7846948160 (J)
2014-02-31 16:51:49.138476  - Current Power 30550.00 (W) Average Power 30498.40 (W) Peak Power 31024.00 (W) Accum Energy 7847102066 (J)
2014-02-31 16:51:54.143891  - Current Power 30237.00 (W) Average Power 30429.35 (W) Peak Power 30837.00 (W) Accum Energy 7847254243 (J)
2014-02-31 16:51:59.149293  - Current Power 30329.00 (W) Average Power 30525.75 (W) Peak Power 31026.00 (W) Accum Energy 7847407330 (J)
```

Figure 4. xtpget -c 15 -d 5 -w 20

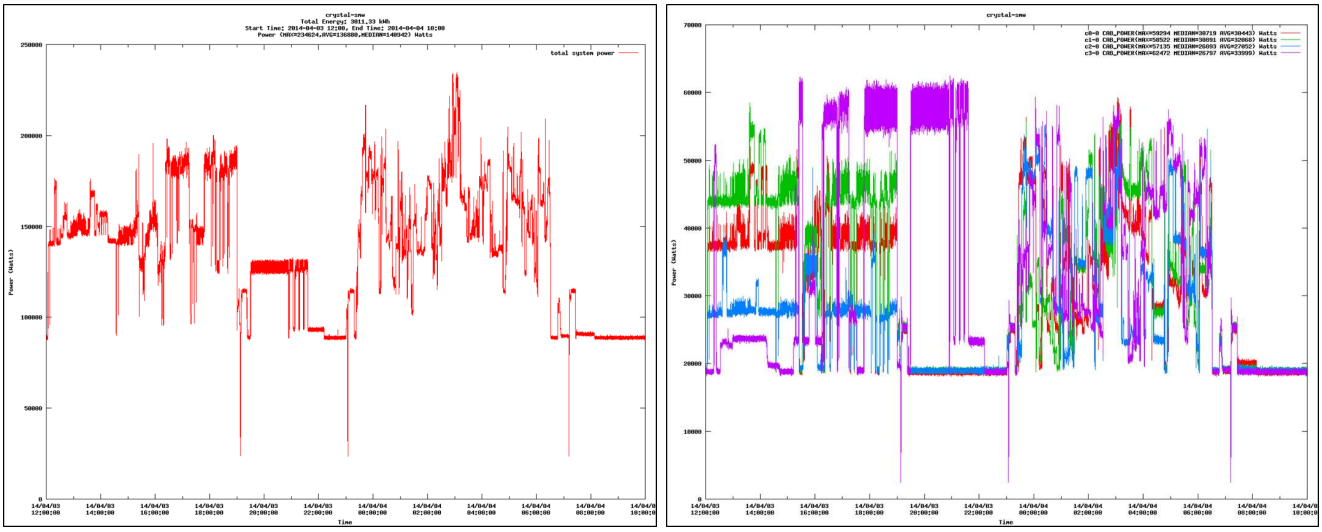

Figure 5. Total System Power (left) and Cabinet Power (right) power plots. 22 hours of four cabinet system data from PMDB's pmdb.cc_data table

```
crayadm@smw:> cray_pmdb_report_energy_single_job.sh 1348984
  APID   | Joules  |          KW/h          |     Runtime
---------+---------+------------------------+-----------------
 1348984 | 1429854 | 0.39718166666666666667 | 00:16:00.331099
(1 row)

 Component  | NID | Joules
------------+-----+--------
 c0-0c2s1n3 | 135 | 254804
 c0-0c2s2n0 | 136 | 259806
 c0-0c2s2n1 | 137 | 249678
 c0-0c2s2n2 | 138 | 249055
 c0-0c2s6n3 | 155 | 264470
 c0-0c2s9n0 | 164 | 152041
(6 rows)
```

```
crayadm@smw:> cray_pmdb_report_energy_single_job.sh 1348989
  APID   | Joules  |          KW/h          |     Runtime
---------+---------+------------------------+-----------------
 1348989 | 1406221 | 0.39061694444444444444 | 00:16:01.975384
(1 row)

 Component  | NID | Joules
------------+-----+--------
 c0-0c2s0n0 | 128 | 264193
 c0-0c2s0n1 | 129 | 248445
 c0-0c2s0n2 | 130 | 250200
 c0-0c2s0n3 | 131 | 254549
 c0-0c2s1n0 | 132 | 249498
 c0-0c2s1n1 | 133 | 139336
(6 rows)
```

Figure 6. Text based application energy profiling output for apids: 1348984 (left) and 1348989 (right)

The system- and node-level data in the PMDB should help make that effort straightforward. The next step is to select a power cap for each of your compute node types that is slightly higher than the previously measured peak (or at least higher than the average measured active) power consumption of the important applications at your site. After determining what the node-level power capping targets should be, the xtpmaction CLI can be used to create, edit, and apply your settings. One may find that `xtpmaction -a power --interactive` tuning support is useful during this step of the process. The intent of this usage mode is to allow your applications to run at or near maximum performance but also avoid unexpected power spikes. This is likely an iterative process to set up, making small adjustments until the correct balance for a given site is found. There are several variations of this use case that can be considered including

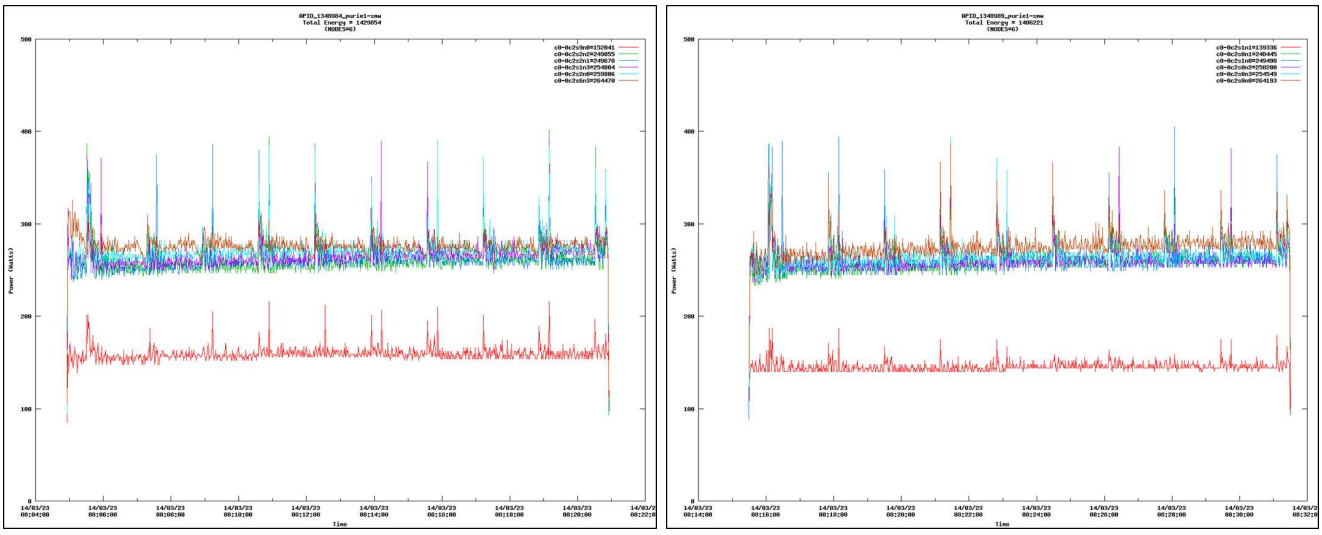

Figure 7. gnuplots of application power for apids: 1348984 (left) and 1348989 (right)

different cap settings for nights, weekends, or particular days of the week.

For sites that may require power cap settings that are more restrictive, the use case is likely very similar to what was just outlined above. The key difference is that aggressive power capping is likely to have a more drastic negative impact on any jobs that when left uncapped would exceed the required cap settings. For these applications it may be possible to mitigate some of the node-to-node performance imbalances caused by power capping by running these applications at a lower frequency with the `aprun --p-state=`*KHz* option.

A third use case for system power capping is in support of events that call for temporary reduction in power consumption. With up-front planning, a site can be prepared by creating one or more named power capping profiles that target different maximum system power limits. The interactive capabilities of xtpmaction mentioned above can also be used to do what-if calculations when generating contingency profiles. This use case is likely to also look at the trade-offs associated with manually powering off some nodes during extremely limiting situations where power capping alone is not practical to run critical jobs with acceptable performance.

We believe that with current XC30 systems all three of these use cases above can be implemented today. However, we also know that there is much more that can be done in this area with the help of third party workload management software, more on that topic in Section IV-A.

E. Application level p-state at job launch

As a demonstration of `aprun --p-state=`*KHz* capability, the graphs in Figure 8 show application energy, and application performance for a series of DGEMM runs on older prototype XC30 hardware. The raw numbers in this data are less important than the clear indication that there are opportunities to save energy using the p-state controls, and that tools are available to collect sensor data to evaluate the results of benchmarks and real application runs.

IV. Proposed new power monitoring and management features

In this section, we will preview feature either in development or planned for development. All features and functionality, dates, and figures specified are preliminary and based on current expectations and are subject to change without notice.

A. Workload manager (external) power monitoring and management interface

Cray is in the process of implementing interfaces that allow third party workload managers (and other authorized software/users) running on select service nodes the ability to monitor and control the power and energy consumption of Cray XC30 systems. At the lowest level, we are implementing RESTful Web Services APIs and JSON-encoded request/response packets. Cray is building an OpenStack [18] style service on top of the low-level APIs called *capmc* (*C*ray *A*dvanced *P*ower *M*onitoring and *C*ontrol). This service and underling APIs is a first step in integrating Cray system monitoring and management with OpenStack components. In addition to implementing the JSON-based RESTful Web Services API in accordance with OpenStack conventions, the OpenStack Keystone [19] service will be integrated to provide authentication (authN) and high-level authorization (authZ) services. The capmc service is intended to allow workload management software (or other authorized users) a convenient layered interface for software development,

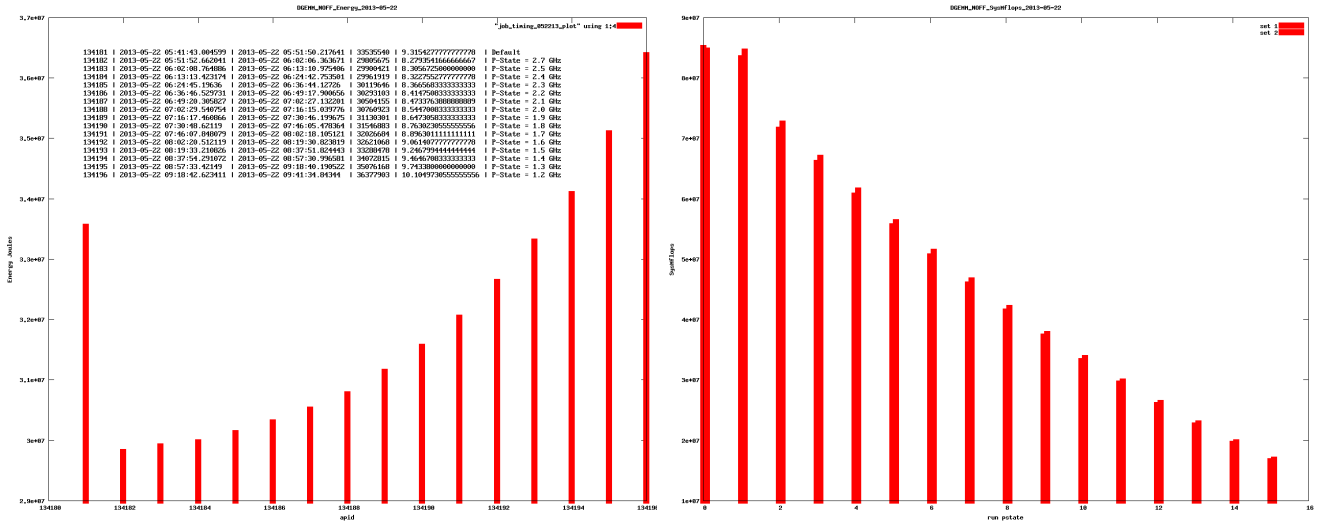

Figure 8. DGEMM Node-level energy and System Mflops for test runs at: Turbo, 2.7 - 1.2 GHz

testing, and integration. We expect that third party workload manager (WLM) vendors and others will be able to use these new capabilities exposed via the capmc service to implement advanced power monitoring and management use cases including: the ability to turn off idle nodes, manage node level power capping, and access power and energy data for the running system, nodes, and applications [20] [21] [22].

The capmc service implements multiple subcommands (sometimes referred to as applets):

1) capmc node_on, node_off, node_rules, and node_status: These subcommands are all related to managing how many compute nodes are powered on or off. The node_on and node_off commands turn power on and boot the nodes into a usable state or cleanly shutdown and power off the requested nodes. The *node_on* and *node_off* subcommands will be implemented as non-blocking operations, in that the service will complete after communication of the request to the system. These subcommands will only fail if that communication fails or invalid parameters are detected.

The node_rules subcommand informs the third party software about hardware (and perhaps site-specific) rules and timing constraints that allow for efficient and effective management of idle node resources. The data returned by the node_rules command will inform the caller how long on and off operations should be expected to take, the minimum amounts of time nodes should be left off to actually save energy, and perhaps limits on the number of nodes that should be turned on or off at any given time to prevent rapid changes is system power consumption on platforms enforcing ramp-rate-limiting. While the node_status subcommand is not limited to use when managing idle nodes, it provides third party software the ability to validate that requested node_on or node_off operations have completed. Note that this ability to check on the status of nodes after performing asynchronous operations is not required but expected to be useful.

2) get_system_power, and get_system_power_details: The *get_system_power* subcommand is used to access minimum, average, and maximum power for the system for a specified window of time. The caller is expected to supply a starting time and window length. The *get_system_power_details* subcommand is used to request minimum, average, and maximum power data for each of the cabinets in the system. As the get_system_power subcommand, the caller of get_system_power_details is expected to provide a starting time and time window length. The ability to access historical data is limited by the size of the system and the amount of resources dedicated to the backing database (PMDB). These calls are also expected to limit the maximum valid time window to one hour. It is expected that third party WLM software will poll for this type of system power data on each scheduling cycle, on a one to five minute timer, or on-demand from an interactive system workload administrator.

3) get_node_energy_counter, get_node_energy_stats, and get_node_energy: The *get_node_energy_stats*, *get_node_energy*, *get_node_energy_counter* allow flexible access to node energy data by nid-list, job-id, or ALPS application-id (apid). For these commands the flexibility starts with the option of supplying an apid that the system-service can then use to generate node-list, start-time and end-time information. The caller can also use an apid with an explicit start-time and end-time options to get information on a running application.

The *get_node_energy_stats* subcommand returns total energy for the selected nodes, average energy for nodes in

the set, standard deviation of energy for nodes in the set, two ordered pairs of (NID, energy) for the minimum and maximum energy nodes, the duration of the interval in seconds, and the node count. All energy values are in joules. This output format is expected to be very useful and efficient when dealing with large node counts, as it can fully leverage the capabilities of the system-service database to generate statistics.

The get_node_energy subcommand takes the same user inputs as get_node_energy_stats but rather than returning statistics it returns the time window in seconds for the data returned, the total node count, and an array of node/energy data with one element for each selected node. This output format will scale with the number of selected nodes, and allows the caller more flexibility in processing the energy data.

The *get_node_energy_counter* subcommand requires an explicit point in time and does not calculate energy used over a time window like the previous two subcommands. The data returned by get_node_energy_counter is the raw accumulated energy counter value for each selected node. The raw accumulated energy (snapshot) data for any given node is only useful when compared to another snapshot for the same node. This command places the most amount of work in the hands of the caller but allows for the most flexibility. Using this call third party software can track total energy, and running energy usage of long running applications, where the runtime of the application may be longer then the depth of data in the system-service database. This also would allow the third party WLM software to directly deal with other advanced use cases like suspend/resume, job migration, etc. Note that this interface can not be used to access data at granularity of less then one second.

4) get_power_cap_capabilities and get_power_cap, set_power_cap: The *get_power_cap_capabilities* and *get_power_cap, set_power_cap* subcommands allow for third party software management of node level power capping. The high level goal of these three subcommands is to enable flexible, efficient node-level (and finer when available) capabilities that can support multiple use cases without enforcing policy.

Given a list of nodes, the subcommand get_power_cap_capabilities will return information about power capping capabilities, controls, and valid ranges. These capabilities are returned in a structured way where information is grouped for all cases where the hardware is common. So even though the call may request capabilities for all of the compute nodes in a system, the maximum response size is limited to one group for each hardware node configuration in the system. On current two socket XC30 nodes, there is a single ‘node’ control that will have minimum/maximum range information in watts. Cray XC30 nodes with accelerators have an additional ‘accel’ control that also has minimum/maximum range information in watts. This infrastructure is an extension of the static system power capping introduced in section II-F.

The get_power_cap subcommand will return the power-capping control(s) and current settings for all requested compute nodes. Note that a power-cap setting (value) of 0 is a special case for *‘not-capped’*. The set_power_cap subcommand allows the authorized caller to set the same controls that are returned by get_power_cap within the minimum/maximum constraints returned by get_power_cap_capabilities. It is expected that third party software will interact with these three power-capping subcommands using formatted JSON.

One way to envision using the node level power capping would be to provision multiple batch queues, where each queue is assigned a different node-level power cap. The number of nodes assigned to each queue could be either static, or dynamic but for simplicity we will consider the static use case. The power cap settings and the number of nodes assigned to the queues can then be used to limit total worst case system power draw. In this configuration we would expect that there might be policies put in place that reward use of the lower power queues. The WLM would use capmc’s power capping subcommands at queue initialization time, and whenever static (or dynamic) adjustments in the number of nodes assigned to queues are made. Another use case would allow users to indicate a desired per-node power reservation at job submission to the WLM. The WLM would in turn use the requested total power for the job as a scheduling constraint, and at job launch time, the WLM would use the capmc set_power_cap subcommand to configure the node-level constraints for all of the nodes assigned to the job.

The ability for WLM to monitor and manage system and node level power makes several new use cases available and should allow system to run much closer to site power limits while maximizing total system performance. Cray has started working with WLM vendors to communicate the proposed capmc APIs; vendors in turn are expected to start evaluating how these new capabilities might fit into their product roadmaps.

B. Unified PMDB + SEDC Database

Cray has plans in place to modify SEDC software to use the PMDB as its backing store. This change will allow for much easier access to data with standard SQL queries from the SMW, as well as enable use cases that integrate access to SEDC sensors with the high-speed data currently collected and stored in the PMDB. The current plan will add two new sensor data tables to the PMDB, one for data collected at the blade-level and the other for data collected at the cabinet level. This mirrors the current PMDB support for blade and cabinet level data collection and table partitioning. The SEDC work is also targeting improvements in SEDC data configuration management.

C. Moving PMDB off-SMW

Cray is investigating options for moving the PMDB off of the SMW. Motivations for moving the database off-SMW include the ability to scale to very large system sizes while retaining node-level data for long periods of time and decoupling database-related load from the SMW. Another motivation is a potential for more direct access to the data than is available on the SMW. At this point, we expect that continuing to run the PMDB on the SMW is a good option for many site configurations. With that in mind, this work will also consider other possible on-SMW PMDB hardware and software configurations.

V. Conclusion

In this paper we have highlighted the current power monitoring and management capabilities available to customers using Cray XC30 systems. We have described and partially demonstrated some basic use cases for current functionality, as well as providing some outlook and information about future power monitoring and management features in the development and planning stages. It is our intent for this information to be useful in promoting ongoing dialogue between the Cray Power Management team and customers interested in HPC system and application power monitoring, management, and efficiency.

Acknowledgments

We would like to thank all of engineers, testers and others who have worked on Cray XC30 power monitoring and management. Without their hard work and attention to detail, this paper would not have been possible. We would also like to thank the customers who have encouraged us to provide enhanced power monitoring and control capabilities. Ongoing customer input and dialogue is critical to our future success.

References

[1] "US Department of Energy: Office of science, exascale challenges," (Accessed 11.Apr.14). [Online]. Available: http://science.energy.gov/ascr/research/scidac/exascale-challenges

[2] "Green500: The green 500 list," (Accessed 11.Apr.14). [Online]. Available: http://www.green500.org

[3] J. H. Laros, III, "Measuring and tuning energy efficiency on large scale high performance computing platforms," (Accessed 11.Apr.14). [Online]. Available: http://www.sandia.gov/~jhlaros/publications/SAND_5702.pdf

[4] "EE HPC Working Group," (Accessed 11.Apr.14). [Online]. Available: http://eehpcwg.lbl.gov

[5] "EE HPC Working Group: Energy efficiency considerations for hpc procurement documents," (Accessed 11.Apr.14). [Online]. Available: http://docs.google.com/viewer?a=v&pid=sites&srcid=bGJsLmdvdnxlZWhwY3dnfGd4OjY1ZjRjNDE2ZTZhY2I3Njk

[6] "PostgreSQL," (Accessed 11.Apr.14). [Online]. Available: http://www.postgresql.org/

[7] "Monitoring and managing power consumption on the Cray XC30 system, Cray publication S-0043-72," (Accessed 11.Apr.14). [Online]. Available: http://docs.cray.com/books/S-0043-72/S-0043-72.pdf

[8] "Managing system software for the Cray Linux Environment, Cray publication S-2393-52xx," (Accessed 11.Apr.14). [Online]. Available: http://docs.cray.com/books/S-2393-52xx/S-2393-52xx.pdf

[9] "PAPI: Performance Application Programming Interface," (Accessed 11.Apr.14). [Online]. Available: http://icl.cs.utk.edu/papi/

[10] "rapl(5), Cray Inc., Performance Tools 6.1.4 man pages," (Accessed 11.Apr.14). [Online]. Available: http://docs.cray.com/cgi-bin/craydoc.cgi?mode=Show;q=PAPI%20RAPL;f=man/xt3_patm/61/cat5/rapl.5.html

[11] "xtpmaction(8), Cray Inc., System Management Workstation (SMW) 7.2.UP00 man pages," (Accessed 11.Apr.14). [Online]. Available: http://docs.cray.com/cgi-bin/craydoc.cgi?mode=Show;q=;f=man/smwm/72/cat8/xtpmaction.8.html

[12] "Intel Intelligent Power Node Manager," (Accessed 11.Apr.14). [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/technical-specifications/intelligent-power-node-manager-specification.pdf

[13] "Intel 64 and IA-32 architectures software developer's manual, volume 3b: System programming guide, part 2, order number: 253669-047US," (Accessed 11.Apr.14). [Online]. Available: http://download.intel.com/products/processor/manual/253669.pdf

[14] J. H. Laros, III, K. T. Pedretti, S. M. Kelly, W. Shu, and C. T. Vaughan, "Energy based performance tuning for large scale high performance computing systems," in *Proceedings of the 2012 Symposium on High Performance Computing*, ser. HPC '12. San Diego, CA, USA: Society for Computer Simulation International, 2012, pp. 6:1–6:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2338816.2338822

[15] H. Poxon, "New functionality in the Cray performance analysis and porting tools," in *Proc. Cray User Group (CUG) conference*, Lugano, Switzerland, May 2014.

[16] G. Fourestey, B. Cumming, and L. Gilly, "First experiences with validating and using the Cray power management database tool," in *Proc. Cray User Group (CUG) conference*, Lugano, Switzerland, May 2014.

[17] A. Hart *et al.*, "User-level power monitoring and application performance on Cray XC30 supercomputers," in *Proc. Cray User Group (CUG) conference*, Lugano, Switzerland, May 2014.

[18] "OpenStack," (Accessed 11.Apr.14). [Online]. Available: http://www.openstack.org

[19] "OpenStack Keystone," (Accessed 11.Apr.14). [Online]. Available: http://docs.openstack.org/developer/keystone/

[20] “Moab Workload Manager, Administrator Guide 7.2.7, February 2014, Adaptive Computing,” (Accessed 11.Apr.14). [Online]. Available: http://docs.adaptivecomputing.com/mwm/7-2-7/mwmAdminGuide-7.2.7.pdf

[21] “SLURM: Power saving guide,” (Accessed 11.Apr.14). [Online]. Available: http://slurm.schedmd.com/power_save.html

[22] “PBS Professional’s Green Provisioning,” (Accessed 11.Apr.14). [Online]. Available: http://www.pbsworks.com/Solution.aspx?lev=1&id=10