Cray XC40 Power Monitoring and Control for Knights Landing

Steven J. Martin, David Rush
Cray Inc.
Chippewa Falls, WI USA
{*stevem,rushd*}*@cray.com*

Matthew Kappel, Michael Sandstedt, Joshua Williams
Cray Inc.
St. Paul, MN USA
{*mkappel,msandste,jw*}*@cray.com*

***Abstract*—This paper details the Cray XC40 power monitoring and control capabilities for Intel Knights Landing (KNL) based systems. The Cray XC40 hardware blade design for Intel KNL processors is the first in the XC family to incorporate enhancements directly related to power monitoring feedback driven by customers and the HPC community. This paper focuses on power monitoring and control directly related to Cray blades with Intel KNL processors and the interfaces available to users, system administrators, and workload managers to access power management features.**

***Keywords*-Power monitoring; power capping; RAPL; energy efficiency; power measurement; Cray XC40; Intel Knights Landing**

I. Introduction

Cray has provided advanced power monitoring and control capabilities on XC-class systems starting with the release of System Management Workstation (SMW) 7.0.UP03 and Cray Linux Environment (CLE) 5.0.UP03 in June of 2013. That first release had support for the first XC blades, equipped with Intel Xeon Sandy Bridge (E5-26XX) processors. In subsequent releases, Cray has delivered additional features, and support for blades with successive generations of Intel Xeon processors, Xeon-Phi Coprocessors, and NVIDIA GPU accelerators. Cray customers and research scientists have published papers about Cray's power monitoring and control capabilities [1] [2], and are utilizing these capabilities in their research [3] [4] [5] [6] [7] [8] [9].

The Cray hardware blade designed for Intel KNL processors is the first in the XC family to incorporate enhancements directly related to power monitoring feedback driven by customers and the HPC community [10] [11] [12] [13] [14] [15].

In previous work [16] [17] [18] Cray advanced power management development team members have outlined Cray XC30 power management capabilities and CAPMC Workload Manager (WLM) interfaces.

In this work, we detail enhancements to power monitoring and control. Improvements in monitoring include increased sampling rates, higher fidelity sensors, and the ability to break-out node point-in-time power telemetry into CPU and memory components. These improvements allow for deeper analysis of power/performance in HPC codes at a finer-grain than previously possible.

This paper is organized as follows: In section II, we detail blade-level Hardware Supervisory System (HSS) architecture changes introduced to enable improved telemetry gathering capabilities on Cray XC40 blades with Intel KNL processors. Section III provides information on updates to interfaces available for power monitoring and control that are new for Cray XC40 systems and the software released to support blades with Intel KNL processors. Section IV shows monitoring and control examples.

II. Enhanced HSS Blade-Level Monitoring

The XC40 KNL blade incorporates a number of enhancements over previous XC blade types that enable the collection of power telemetry in greater detail and with improved accuracy. Previous XC blade types have used the Blade Micro to poll sensors from a single I2C bus operating at 100KHz. As can be seen in figure 1, on the XC40 KNL blade, this function has been moved to several parallel buses fanning out from the KNL Processor Daughter Card Level 0 Compute (KLOC) FPGAs and polled by the Blade Controller (BC) directly.

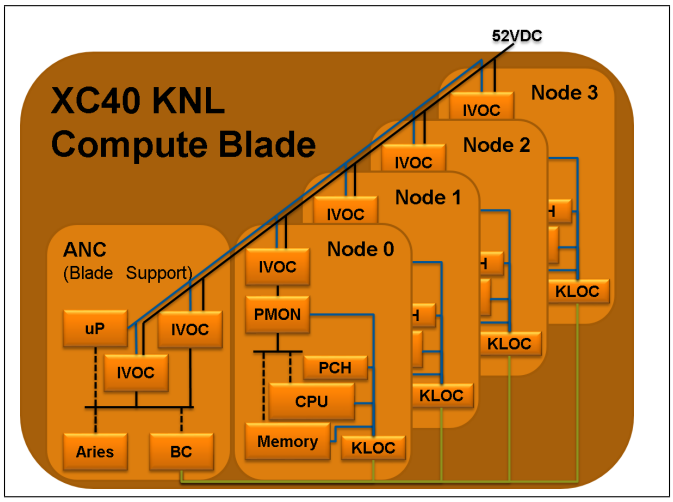

Figure 1: KNL High-Level Power Monitoring

With the improved parallel topology and increased bus speeds, more sensors can be polled, and can potentially be polled at higher frequencies. Figure 2 shows a more detailed mapping of devices to I2C masters in the XC40 KNL blade.

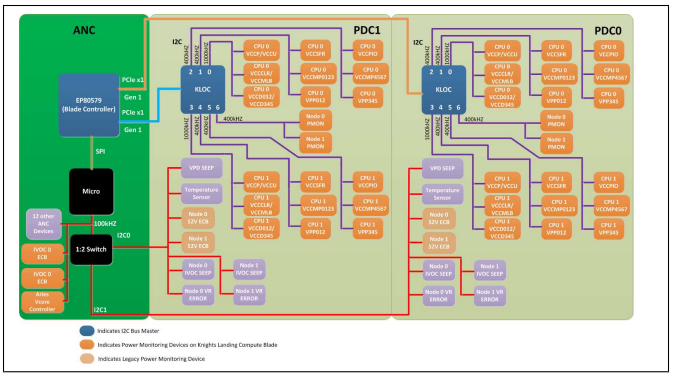

Figure 2: KNL I2C Power Monitoring

The node-level power sensor, or PMON as it's referenced in figures 1 and 2, is a Texas Instruments LM5056A [19]. With a 12-bit ADC and 1KHz sampling rate, this part greatly improves the accuracy of node-level power telemetry over that available from the 8-bit ADC used with previous XC blade types. Additionally, the TI LM5056 provides a hardware averaging filter, as schematically represented in figure 3. With hardware averaging enabled, the quality of polled telemetry greatly benefits from the 1KHz sampling rate, even when polled at a lower frequency. Cray's polling implementation adjusts the length of the filter dynamically to match the BC's polling interval, ensuring that derivative telemetry, as much as possible, is arithmetically equivalent to that which would be produced with 1KHz polling.

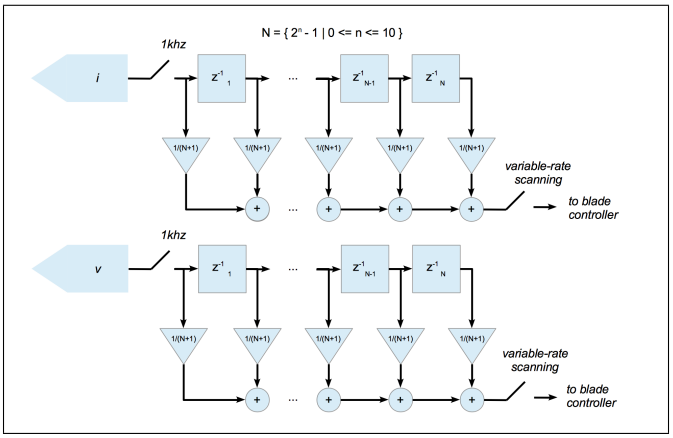

Figure 3: TI LM5056 Hardware Averaging Filter

In addition to node-level telemetry, the XC40 KNL blade provides telemetry from each of the nodes' DC-DC converters. The power regulator logical layout for KNL Processor Daughter Card (KPDC) physical node position 0 is detailed in figure 4. The color coding in the figure denotes CPU-related power regulators in blue and memory-related regulators in green.

Leveraging the mutli-bus I2C layout, the blade controller can more efficiently poll voltage, current, power and

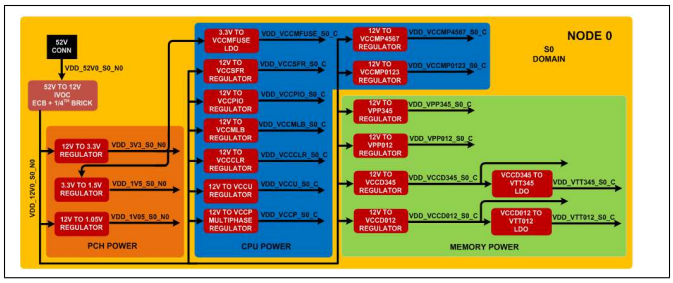

Figure 4: KPDC-NODE0 Power Regulation Layout

temperature from all of the DC-DC converters. Power is polled from the converters at a rate of 10Hz, and from this data, aggregate CPU-domain and memory-domain power telemetry are published both locally on the blade controller and in-band in pm_counters (see section III). Voltage, current, power and temperature are available from each converter individually at scan rates up to 1Hz via System Environment Data Collections (SEDC) [20]. SEDC data is transmitted over the HSS out-of-band network to the Cray Power Management Database (PMDB) and stored in the *pmdb.bc_sedc_data* table. The Cray PM tutorial slides for CUG2015 [21] can be a valuable resource on SEDC, PMDB and other Cray XC40 related power management subjects.

XC40 KNL blades also have secondary node-level power sensors physically situated on the nodes' IVOC 52V-12V converters. Figure 5 shows an isometric view of a KPDC (of which each blade has two, giving a total 4 nodes per blade). An IVOC for each of the physical node locations 0 and 1 can be seen in the figure.

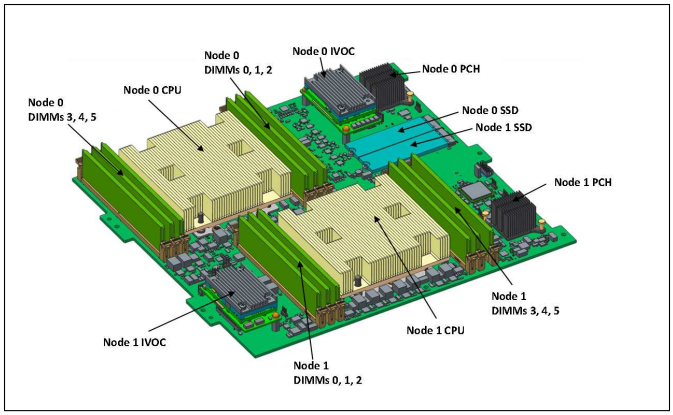

Figure 5: KPDC Isometric View

IVOC power sensing is provided by a Texas Instruments LM5066I controller [22], very similar in measurement capability to the LM5056 PMON. To ensure accuracy, the LM5066I power sensor undergoes per-unit calibration during the IVOC manufacturing process. The LM5056 PMON is in turn dynamically calibrated against the IVOC LM5066I power sensor at runtime. By this method, published node-level power telemetry is calibrated on a per-node basis.

The PMON calibration takes advantage of the TI LM5066I power sensor's hardware energy accumulator to maximize accuracy with minimal computational burden for the BC. Polling the LM5066I energy accumulator approximately once every thirty seconds, the BC can compute calibrated average power draw over arbitrary time periods. The blade controller compares these calibrated average power readings to measurements from the LM5056 PMON over corresponding periods. As shown in figure 6, these comparisons are used to correct subsequent PMON readings.

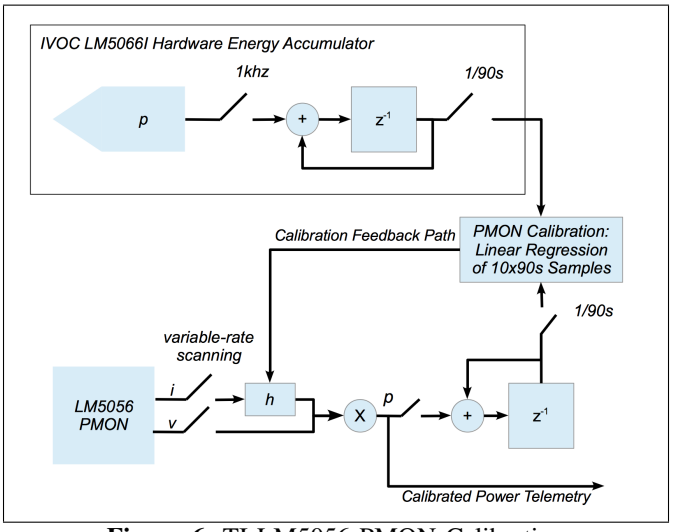

Figure 6: TI LM5056 PMON Calibration

The period used for collecting calibration samples is 90 seconds. Amortizing timing discrepancies over the 90-second averaging period ensures readings can be compared with a high degree of confidence, leading to an accurate calibration. The calibration transfer function (*h* in figure 6) is constructed by performing a linear regression on the ten most recent 90-second sample pairs. The calibration transfer function is used for all published current and power readings, thus minimizing inaccuracies introduced by unit-to-unit sense resistor variation.

III. Power Monitoring and Control Interface Updates

This section identifies power monitoring and control related interface updates that will be released along with support for the Cray XC40 blades with Intel KNL processors. This section is not intended to be a complete reference for all Cray XC power management interfaces.

A. SEDC

Select NODE0 SEDC scan IDs for the Cray XC40 blade with Intel KNL processors are listed in Figures 7, and 8. Corresponding scan ids are available for each of the four logical nodes on the blade.

ID	Name
2128	BC_P_NODE0_VCCLR_POUT
2129	BC_P_NODE0_VCCMLB_POUT
2130	BC_P_NODE0_VCCD012_POUT
2131	BC_P_NODE0_VCCD345_POUT
2132	BC_P_NODE0_VCCP_POUT
2133	BC_P_NODE0_VCCU_POUT
2134	BC_P_NODE0_VCCSFR_POUT
2135	BC_P_NODE0_VCCMP0123_POUT
2136	BC_P_NODE0_VPP012_POUT
2137	BC_P_NODE0_VCCPIO_POUT
2138	BC_P_NODE0_VCCMP4567_POUT
2139	BC_P_NODE0_VPP345_POUT

Figure 7: NODE0 SEDC VRM power sensors

ID	Name
1888	BC_T_NODE0_VCCLR_NTC
1889	BC_T_NODE0_VCCMLB_NTC
1890	BC_T_NODE0_VCCD012_NTC
1891	BC_T_NODE0_VCCD345_NTC
1892	BC_T_NODE0_VCCP_NTC
1893	BC_T_NODE0_VCCU_NTC
1894	BC_T_NODE0_VCCSFR_DIE
1895	BC_T_NODE0_VCCMP0123_DIE
1896	BC_T_NODE0_VPP012_DIE
1897	BC_T_NODE0_VCCPIO_DIE
1898	BC_T_NODE0_VCCMP4567_DIE
1899	BC_T_NODE0_VPP345_DIE

Figure 8: NODE0 SEDC VRM thermal sensors

B. Cray pm_counters

Cray pm_counters are an in-band interface that provides power and energy information to applications and system software running on compute nodes. The actual data is collected out-of-band on HSS controllers and made available on compute nodes in the sysfs directory: **/sys/cray/pm_counters/**.

This directory contains a set of files, sometimes referred to as "descriptors" that can be read by anyone. A few example descriptors:

- **power**: Point in time power, watts
- **energy**: Accumulated energy, joules
- **power_cap**: Power cap, watts
- **generation**: Number of times power cap has changed
- **raw_scan_hz**: Current sample rate (default is 10 Hz)

System with accelerators have the following three additional descriptors:

- **accel_power**: Accelerator point in time power, watts
- **accel_energy**: Accelerator accumulated energy, joules

- **accel_power_cap**: Accelerator power cap, watts

The rate is at which all data is sampled is made available through the **raw_scan_hz** descriptor. The default sample rate is 10 Hz but the enhanced HSS blade-level monitoring may allow for sample rates in excess of 100Hz. We hope to document this enhanced capability on this and future blade types after further testing and validation.

The enhanced HSS blade-level monitoring adds the following new descriptors to the pm_counters interface:

- **cpu_power**: CPU point in time power, watts
- **cpu_energy**: CPU accumulated energy, joules
- **memory_power**: Memory point in time power, watts
- **memory_energy**: Memory accumulated energy, joules

These new descriptors are not supported on nodes introduced prior to the new XC40 blade with Intel KNL processors. Like the accel_* descriptors these will only be visible on supported nodes. Future XC40 blades are expected to support these new memory and cpu descriptors.

As an example of using pm_counters, applications could sample the energy descriptor at the beginning of execution and then at the end of execution to determine the total energy that was consumed during the run. A variety of other statistics could be calculated in a similar manner. Note that many of these sorts of computations are already handled for users of Cray XC systems with the Resource Utilization Reporting (RUR) 'energy' plugin which is described in the next section.

More detailed information on pm_counters can be found in Cray supplied online customer documentation [23].

C. The RUR Energy Plugin

Resource Utilization Reporting (RUR) is an administrator tool for gathering statistics on how system resources are being used by applications and jobs. RUR is a low-noise, scalable infrastructure that collects compute node statistics before an application or job runs and again after it completes. The extensible RUR infrastructure allows plugins to be easily written to collect and report a variety of data. Plugins can log their output to a number of configurable locations including users home directories on login nodes. On the SMW this data is generally available in the file: **/var/opt/cray/log/partition-current/messages-date**

One such Cray supplied plugin is the 'energy' plugin. It collects compute node power and energy related usage data on behalf of application users. Much of the data used by the RUR energy plugin comes from pm_counters as described in the last section.

A small sampling of RUR energy plugin data:

- **energy_used**: Total energy used across all nodes, joules
- **nodes**: Number of nodes in the job
- **nodes_power_capped**: Number of nodes power capped
- **nodes_throttled**: Number of nodes throttling
- **max_power_cap**: Maximum nonzero power cap, watts
- **min_power_cap**: Minimum nonzero power cap, watts

As with pm_counters, enhanced blade-level monitoring also adds the following new RUR energy data:

- **cpu_energy_used**: Total CPU energy, joules
- **memory_energy_used**: Total memory energy, joules

All of the above data is cumulative for all nodes that the job runs on. The RUR energy plugin can also be configured in a "verbose" mode which will additionally generate an RUR energy plugin log for each node in the job. For large jobs this will generate a substantial number of logs. The data in each verbose log entry is different from the previous descriptions. Please reference the appropriate documentation to learn more about them.

More detailed information on the RUR energy plugin can be found in the Cray supplied online customer documentation [24].

D. C-State and P-State Limiting

Processor C-States are essentially hardware power savings states. They define the degree to which a processor is sleeping, or powered down. The shallowest C-state indicates normal operation where instructions are being executed. All other C-States below that indicate deeper sleep states with more of the processor complex powered down. If the current workload is light, the processor will be found in the deeper C-States more frequently and for greater durations. An idle processor will transition towards the deepest enabled C-State. The deeper the C-State, the higher the latency before the processor can start executing instructions. Processor C-State limiting allows the administrator (or workload manager) to set the maximum C-State depth that a processor can descend into. C-State limiting can aid in setting a floor on both wake-up latency and power consumption. C-State limiting may increase power consumption but looking into the future it may play a part in controlling power ramp rates on large systems.

Processor P-States on the Cray XC represent the frequencies that a processor can run at. Users can request specific frequencies via ALPS or WLM command line options. If no P-State request has been made, a processor will run at the default frequency which is the maximum possible including any available turbo. P-State limiting allows an administrator to set upper and lower bounds on the frequencies that a processor can run at. The actual operating frequency will be constrained to be within this range and any request outside of it will be raised or lowered so that it adheres to the limits. P-State limiting can aid in setting ceilings and floors on point-in-time power consumption, and in managing runtime application performance variability.

C-State and P-State limiting is a new feature in CLE 6.0, the first release supporting Intel KNL processors. The capmc

command provides the interfaces for using it through a series of new applets available to workload managers:

- **get_sleep_state_limit_capabilities**:
  Returns all valid C-States for target node(s)
- **get_sleep_state_limit**:
  Returns the current C-State limits for target node(s)
- **set_sleep_state_limit**:
  Sets the C-State limit for the target node(s)
- **get_freq_capabilities**:
  Returns all valid P-States for target node(s)
- **get_freq_limit**:
  Returns the current P-State limits for target node(s)
- **set_freq_limit**:
  Sets the P-State limits for the target node(s)

More detailed information on these capmc applets can be found in the Cray online customer documentation [18].

IV. KNL Power Monitoring and Control Examples

A. OUT-Of-Band: KNL Job Reports (text)

There are several example scripts that ship with the Cray SMW on XC40 systems. They can by found in */opt/cray/hss/default/pm/script_examples*. Figure 9 shows the use of an example script used to collect data for one of six four-node test runs. The text report data in figure 10 is for the same test program run, but this report script generates more per node information. This output was edited by replacing actual minimum and maximum power data with **XXX**.

The data in figure 11 shows summary energy and runtime data for the same set of six four-node test runs. Looking at this data we can see that the run for Application ID (APID) 1918 has the lowest energy to solution, while the run for APID 1922 has the fastest runtime. The data shown in figure 11 was created by hand editing output from multiple runs of *cray_pmdb_report_energy_single_job.sh*, but could easily be created by a custom script.

```
 APID | Joules  | KW/h  |  Runtime
------+---------+-------+-------------
 1917 | 1286025 | 0.357 | 00:36:28.98

  Component  | NID | Joules
-------------+-----+--------
 c0-0c0s13n0 |  52 | 334490
 c0-0c0s13n1 |  53 | 317786
 c0-0c0s13n2 |  54 | 318631
 c0-0c0s13n3 |  55 | 315118
```

Figure 9: Text data for APID 1917

```
     |         | max | min | average
 NID | Joules  | (W) | (W) |  (W)
-----+---------+-----+-----+--------
 52  | 334490  | XXX | XXX | 152.86
 53  | 317786  | XXX | XXX | 145.23
 54  | 318631  | XXX | XXX | 145.70
 55  | 315118  | XXX | XXX | 144.09

    cname    | joules
-------------+--------
 c0-0c0s13n3 | 315118
 c0-0c0s13n1 | 317786
 c0-0c0s13n2 | 318631
 c0-0c0s13n0 | 334490

 APID | Joules  | KW/h   |  Runtime
------+---------+--------+------------
 1917 | 1286025 | 0.3572 | 00:36:28.99
```

Figure 10: Text data for APID 1917

```
 APID | Joules  | Runtime
------+---------+------------
 1917 | 1286025 | 36:28.99
 1918 | 1257640 | 34:07.17 << Min Energy
 1919 | 1268345 | 32:42.08
 1920 | 1298037 | 31:58.58
 1921 | 1333215 | 31:43.99
 1922 | 1353328 | 28:52.91 << Fastest
```

Figure 11: Summary text data (APIDs 1917 - 1922)

B. Out-Of-Band: KNL Job Power Plots

The data stored in the Power Management Database (PMDB) on Cray XC40 systems can be used to create graphic power profiles using relatively simple scripts and plotting programs. Figures 12, and 13 show data for the same test runs detailed in IV-A above. Figure 12 plots node, cpu, and memory point-in-time power information for APID 1917. As noted earlier in this paper the cpu and memory accumulated energy and point-in-time power are new data points in the PMDB starting with support for KNL nodes. Information for all six tests runs are presented. The plots have been edited to remove detailed power information.

C. In-Band pm_counters

The data in the */sys/cray/pm_counters* sysfs directory is accessible to any running application. To illustrate how simple it is to explore the data made available, the example in figure 14 uses *grep* to dump the file names and contents:

As previously noted in this paper, the *cpu_energy*, *cpu_power*, *memory_energy*, and *memory_power* descriptors are new and only supported on KNL nodes at this time. The

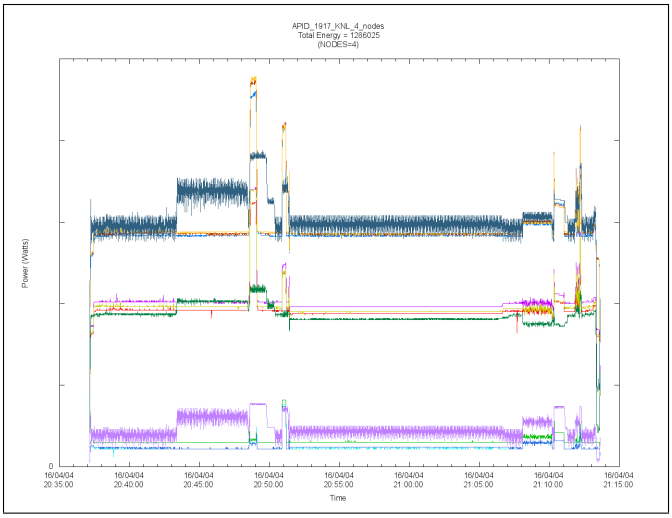

Figure 12: APID 1917 test run plot

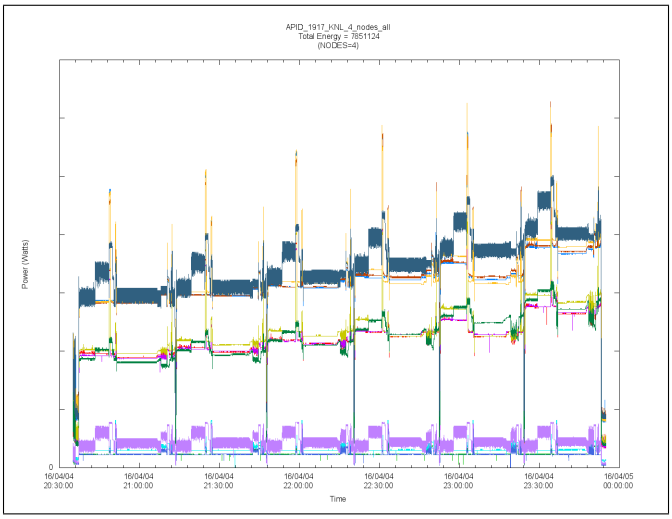

Figure 13: Node, CPU, and Memory plot (APIDs 1917 - 1922)

raw_scan_hz descriptor is also new with the software release that supports KNL nodes, but will also be supported for older blade types. That said, all older blade types are not expected to ever support update frequencies other than 10Hz.

D. The RUR Energy Plugin

Figures 15 and 16 are examples of RUR energy plugin log output obtained for two of the six test runs used in previous examples. Some of the RUR output examples have been edited to show only fields with non-zero data. Note that in this run, three of the four nodes are reported to have been "throttled" where in the lower power run (APID 1917) no nodes reported throttling.

Please see Cray RUR online customer documentation [24] for a full description of all data fields above.

```
nid00052:~ #  cd /sys/cray/
nid00052:~ #  grep -v "FOO" *
pm_counters/cpu_energy:21426998 J
pm_counters/cpu_power:24 W
pm_counters/energy:37726662 J
pm_counters/freshness:3498963
pm_counters/generation:45
pm_counters/memory_energy:5191469 J
pm_counters/memory_power:5 W
pm_counters/power:49 W
pm_counters/power_cap:0 W
pm_counters/raw_scan_hz:10
pm_counters/startup:1459443886386935
pm_counters/version:2
```

Figure 14: Dump of /sys/cray/pm_counters/* .

```
[RUR@34] uid: 12795, apid: 1917,
jobid: 0, cmdname: ./test,
plugin: energy {
  "nodes_throttled":              0,
  "memory_energy_used":           138156,
  "min_accel_power_cap_count":    0,
  "nodes_with_changed_power_cap":0,
  "max_power_cap_count":          0,
  "energy_used":                  1285795,
  "max_power_cap":                0,
  "nodes_memory_throttled":       0,
  "accel_energy_used":            0,
  "max_accel_power_cap_count":    0,
  "nodes_accel_power_capped":     0,
  "min_power_cap":                0,
  "max_accel_power_cap":          0,
  "min_power_cap_count":          0,
  "min_accel_power_cap":          0,
  "nodes_power_capped":           0,
  "nodes":                        4,
  "cpu_energy_used":              846865,
  "nodes_cpu_throttled":          0
}
```

Figure 15: RUR energy plugin output APID 1917

E. C-State and P-State Limiting

The Cray capmc command outputs data in JavaScript Object Notation (JSON) format. The following examples show a minimal subset of the actual JSON output in order to illustrate the core concepts as clearly as possible. For a complete description of capmc JSON syntax please refer to Cray online documentation [18]. The data in the examples were obtained on nodes with Intel Xeon processors instead of nodes with Intel KNL processors in order to avoid sharing too much information before the official product launch.

```
[RUR@34] uid: 12795, apid: 1922,
jobid: 0, cmdname: ./test,
plugin: energy {
  "nodes_throttled":     3,
  "memory_energy_used":  112482,
  "energy_used":         1353047,
  "nodes":               4,
  "cpu_energy_used":     964289,
  "nodes_cpu_throttled": 3
}
```

Figure 16: Non-Zero RUR energy plugin output APID 1922

```
$ capmc get_freq_capabilities --nids 68
PWR_Attrs": [ {
    "PWR_AttrName": "PWR_ATTR_FREQ",
    "PWR_AttrValueCapabilities": [
      2101000, 2100000, 2000000, 1900000,
      1800000, 1700000, 1600000, 1500000,
      1400000, 1300000, 1200000
    ], "PWR_ReturnCode": 0
  }
],
```

Figure 17: *capmc get_freq_capabilities –nids 68.*

In figure 17 we query Node ID (NID) 68 to discover the list of valid frequencies for the processors on that node by calling the *capmc get_freq_capabilities* applet. The *PWR_AttrValueCapabilities* array output data shows values ranging from 2.1 GHz (+ turbo) through 1.2 GHz.

```
$ capmc get_freq_limits --nids 68
"PWR_Attrs": [ {
    "PWR_ReturnCode":      0,
    "PWR_AttrName":
            "PWR_ATTR_FREQ_LIMIT_MIN",
    "PWR_AttrValue":       1200000,
    "PWR_TimeSeconds":     1459709317,
    "PWR_TimeNanoseconds": 655607479
  }, {
    "PWR_ReturnCode":      0,
    "PWR_AttrName":
            "PWR_ATTR_FREQ_LIMIT_MAX",
    "PWR_AttrValue":       2101000,
    "PWR_TimeSeconds":     1459709317,
    "PWR_TimeNanoseconds": 655616142
  }
],
```

Figure 18: *capmc get_freq_limits*

The example n figure 18 calls *capmc get_freq_limits –nids 68* to get the current settings for nid 68. The JSON output shows the default frequency limits of 1.2 GHz (minimum) and 2.1 +turbo (maximum) for this node. Next, the example in figure 19 calls *capmc set_freq_limits –nids 68 –max 1900000 –min 1400000* to adjust the maximum and minimum frequency settings to 1.9 GHz and 1.4 GHz respectively (command output not shown).

```
$ capmc set_freq_limits --nids 68 \
                        --max 1900000 \
                        --min 1400000
```

Figure 19: *capmc set_freq_limits*

Finally, the example in figure 20 calls *capmc get_freq_limits –nids 68* to show the settings were updated. As expected, the *PWR_ATTR_FREQ_LIMIT_MIN* and *PWR_ATTR_FREQ_LIMIT_MAX* attributes have been updated to 1.4 GHz and 1.9 GHz respectively. Note that these interfaces are intended for workload managers running with required privilege and security credentials.

```
$ capmc get_freq_limits --nids 68
"PWR_Attrs": [ {
    "PWR_ReturnCode":      0,
    "PWR_AttrName":
            "PWR_ATTR_FREQ_LIMIT_MIN",
    "PWR_AttrValue":       1400000,
    "PWR_TimeSeconds":     1459709371,
    "PWR_TimeNanoseconds": 291660165
  }, {
    "PWR_ReturnCode":      0,
    "PWR_AttrName":
            "PWR_ATTR_FREQ_LIMIT_MAX",
    "PWR_AttrValue":       1900000,
    "PWR_TimeSeconds":     1459709371,
    "PWR_TimeNanoseconds": 291669333
  }
],
```

Figure 20: *set_freq_limits* then review the changes.

It should be noted that the capmc applets above use attribute names defined in [15] and reflect early Cray APM NRE work for Trinity in collaboration with ACES.

F. KNL Power Capping

Figures 21 show the actual capmc output for Intel KNL based XC40 nodes for the given *capmc set_power_cap* calls. In the example, NID 52-55 are first power capped at XXX watts, then the power cap is removed by setting the cap value back to 0.

After setting all four nodes to a power cap of XXX watts, the same test program used throughout this section was run again and was assigned APID 3808. As seen in figure 22, RUR output correctly identifies that all four nodes

```
# Set power cap to XXX Watts
$ capmc set_power_cap -n "52-55" -N XXX
{ "e":0, "err_msg":"" }

# Run test case...

# Clear power cap
$ capmc set_power_cap -n "52-55" -N 0
{ "e":0, "err_msg":"" }
```

Figure 21: *capmc set_power_cap.*

are capped at the same power setting. Note that the power cap setting (XXX watts) used was higher than the expected average power for the test run, but lower (more restrictive) than the expected maximum power level.

```
[RUR@34] uid: 12795, apid: 3808,
jobid: 0, cmdname: ./test,
plugin: energy {
  "nodes_throttled":      3,
  "memory_energy_used":   137373,
  "max_power_cap_count": 4,
  "energy_used":          1282910,
  "max_power_cap":        XXX,
  "min_power_cap":        XXX,
  "min_power_cap_count": 4,
  "nodes_power_capped":   4,
  "nodes":                4,
  "cpu_energy_used":      845739,
  "nodes_cpu_throttled": 3
}
```

Figure 22: Non-Zero RUR energy plugin output APID 3808

The RUR energy plugin results shown in figure 23 represent the next run of the test program after removing the nodes' power cap by calling *capmc set_power_cap -n "52-55" -N 0.*

```
[RUR@34] uid: 12795, apid: 3809,
jobid: 0, cmdname: ./test,
plugin: energy {
  "memory_energy_used": 136580,
  "energy_used":        1272448,
  "nodes":              4,
  "cpu_energy_used":    839229,
}
```

Figure 23: Non-Zero RUR energy plugin output APID 3809

The data compiled in figure 24 shows that power capping the test program at XXX watts caused total energy_used, memory_energy_used, and cpu_energy_used to all increase a small amount along with total test runtime. The analysis of the two runs is inconclusive with/respect to the effect of the XXX watt power cap.

The results show in figures 24 and 25 could be caused by run-to-run variability. However, the data in figures 22 and 23 indicates that the power capped job (APID 3808) was throttled, and the non-capped job (APID 3809) was not.

apid	run time	node energy used	memory energy used	cpu energy used
3808	36:30	1282910	137373	845739
3809	36:14	1272448	136580	839229

Figure 24: Summary data for APID 3808 and 3809 runs.

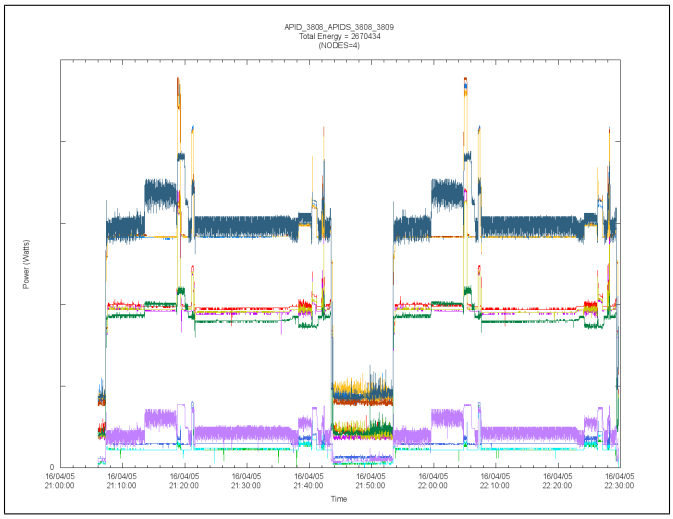

Figure 25: Plot of APIDs 3808 and 3809 running the test program

V. Conclusion

In this paper, we have detailed enhancements to power monitoring and control. Improvements in monitoring include increased sampling rates, higher fidelity sensors, and the ability to break-out node point-in-time power telemetry into CPU and memory components.

At Cray we believe that the ability to monitor and control power and energy is a key to both short and long term success in HPC. Cray has worked closely with Cray customers, Intel, and other vendors to design and test new blades for the Cray XC40 systems featuring Intel Knights Landing processors. That close working relationship has extended to the blade HSS design where Cray has added new monitoring capabilities that improve the ability to collect power and energy data and to make that data available to

system administrators, application developers, and the HPC research community.

REFERENCES

[1] A. Hart, H. Richardson, J. Doleschal, T. Ilsche, M. Bielert, and M. Kappel, "User-level power monitoring and application performance on cray xc30 supercomputers," *Proceedings of the Cray User Group (CUG)*, 2014, (Accessed 31.March.16). [Online]. Available: https://cug.org/proceedings/cug2014_proceedings/includes/files/pap136.pdf

[2] G. Fourestey, B. Cumming, L. Gilly, and T. C. Schulthess, "First experiences with validating and using the cray power management database tool," *arXiv preprint arXiv:1408.2657*, 2014, (Accessed 31.March.16). [Online]. Available: http://arxiv.org/pdf/1408.2657v1.pdf

[3] M. Bareford, "Monitoring the cray xc30 power management hardware counters," *Proceedings of the Cray User Group (CUG)*, 2015, (Accessed 31.March.16). [Online]. Available: https://cug.org/proceedings/cug2015_proceedings/includes/files/pap125.pdf

[4] B. Cumming, G. Fourestey, O. Fuhrer, T. Gysi, M. Fatica, and T. C. Schulthess, "Application centric energy-efficiency study of distributed multi-core and hybrid cpu-gpu systems," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014, pp. 819–829.

[5] K. Pedretti, S. L. Olivier, K. B. Ferreira, G. Shipman, and W. Shu, "Early experiences with node-level power capping on the cray xc40 platform," in *Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing*. ACM, 2015, p. 1.

[6] B. Austin and N. J. Wright, "Measurement and interpretation of microbenchmark and application energy use on the cray xc30," in *Proceedings of the 2nd International Workshop on Energy Efficient Supercomputing*. IEEE Press, 2014, pp. 51–59.

[7] B. Acun, P. Miller, and L. V. Kale, "Variation among processors under turbo boost in hpc systems," (Accessed 31.March.16). [Online]. Available: http://charm.cs.illinois.edu/newPapers/16-08/paper.pdf

[8] S. Byna and B. Austin, "Evaluation of parallel i/o performance and energy with frequency scaling on cray xc30," *Proceedings of the Cray User Group (CUG)*, 2015, (Accessed 31.March.16). [Online]. Available: https://cug.org/proceedings/cug2015_proceedings/includes/files/pap114.pdf

[9] S. Labasan, M. Larsen, and H. Childs, "Exploring tradeoffs between power and performance for a scientific visualization algorithm," in *Large Data Analysis and Visualization (LDAV), 2015 IEEE 5th Symposium on*. IEEE, 2015, pp. 73–80.

[10] "EE HPC Working Group," (Accessed 1.April.16). [Online]. Available: https://eehpcwg.llnl.gov/

[11] "Energy Efficiency Considerations for HPC Procurement Documents (2013)," (Accessed 1.April.16). [Online]. Available: https://eehpcwg.llnl.gov/documents/compsys/ab_procurement_2013.pdf

[12] "Energy Efficiency Considerations for HPC Procurement Documents: 2014," (Accessed 1.April.16). [Online]. Available: https://eehpcwg.llnl.gov/documents/compsys/aa_procurement_2014.pdf

[13] "Trinity/NERSC-8 Use Case Scenarios," (Accessed 1.Apr.15). [Online]. Available: https://www.nersc.gov/assets/Trinity--NERSC-8-RFP/Documents/trinity-NERSC8-use-case-v1.2a.pdf

[14] J. H. Laros, III, S. M. Kelly, S. Hammond, and K. Munch, "Power/Energy Use Cases for High Performance Computing," SANDIA REPORT SAND2013-10789 Unlimited Release Printed December, 2013, Accessed 26.March.15. [Online]. Available: https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/UseCase-powapi.pdf

[15] J. H. Laros, III, D. DeBonis, R. Grant, S. M. Kelly, M. Levenhagen, S. Olivier, and K. Pedretti, "High Performance Computing - Power Application Programming Interface Specification Version 1.0," SANDIA REPORT SAND2014-17061 Unlimited Release Printed August, 2014, Accessed 27.March.15. [Online]. Available: http://powerapi.sandia.gov/docs/PowerAPI_SAND.pdf

[16] S. Martin and M. Kappel, "Cray XC30 Power Monitoring and Management," *Proceedings of the Cray User Group (CUG)*, 2014, (Accessed 28.March.16). [Online]. Available: https://cug.org/proceedings/cug2014_proceedings/includes/files/pap130.pdf

[17] S. Martin, D. Rush, and M. Kappel, "Cray Advanced Platform Monitoring and Control (CAPMC)," *Proceedings of the Cray User Group (CUG)*, 2015, (Accessed 28.March.16). [Online]. Available: https://cug.org/proceedings/cug2015_proceedings/includes/files/pap132.pdf

[18] "CAPMC API Documentation," (Accessed 1.April.16). [Online]. Available: http://docs.cray.com/books/S-2553-11/S-2553-11.pdf

[19] "LM5056/LM5056A High Voltage System Power Measurement Device with PMBus (Rev. A)," (Accessed 7.April.16). [Online]. Available: http://www.ti.com/lit/gpn/lm5056

[20] "System Environment Data Collections (SEDC) Guide," (Accessed 5.April.16). [Online]. Available: http://docs.cray.com/books/S-2491-7204/S-2491-7204.pdf

[21] "Cray XC Power Monitoring and Management Tutorial (Slides)," (Accessed 5.April.16). [Online]. Available: https://cug.org/proceedings/cug2015_proceedings/includes/files/tut102.pdf

[22] "LM5066I 10 to 80 V Hotswap Controller With I/V/P Monitoring and PMBus Interface," (Accessed 7.April.16). [Online]. Available: http://www.ti.com/lit/ds/symlink/lm5066i.pdf

[23] "Monitoring and Managing Power Consumption on the Cray XC System," (Accessed 1.April.16). [Online]. Available: http://docs.cray.com/books/S-0043-7204/S-0043-7204.pdf

[24] "CLE XC System Administration Guide," (Accessed 1.April.16). [Online]. Available: http://docs.cray.com/books/S-2393-5204xc/S-2393-5204xc.pdf