SAND2015-3824C

Exploring MPI Application Performance Under Power Capping on the Cray XC40 Platform

Kevin Pedretti, Stephen L. Olivier, Kurt B. Ferreira
Sandia National Laboratories
Albuquerque, NM, USA
{ktpedre, slolivi, kbferre}@sandia.gov

Galen Shipman
Los Alamos National Laboratory
Los Alamos, NM, USA
gshipman@lanl.gov

Wei Shu
University of New Mexico
Albuquerque, NM, USA
wshu@unm.edu

ABSTRACT

The power consumption of supercomputers has become one of the key bottlenecks limiting performance. Yet current practice in operating these systems does not leverage power management opportunities, instead running at "maximum power" due to the potential impact to application performance and scalability. Many suspect that this will not be sustainable, and that on future systems power will need to be managed as a critical resource and directed to where it has most benefit. Power capping is one promising mechanism for managing power budgets, however its behavior is not well understood for HPC workloads. This paper presents an empirical evaluation of several MPI benchmarks and a full application running under power caps on a Cray XC40 system, one of the first HPC platforms with vendor-supported power capping capabilities. We show that achieving maximum performance under a power cap often requires operating at a lower than default CPU frequency (P-state), and present analysis showing that this is due to cascading slowdowns due to unsynchronized performance variability across nodes made by the underlying power capping mechanism. When operating under a power cap, we observe a performance difference of up to 22.7% between the default and ideal P-state configurations. Our work provides critical analysis and comparison of HPC application performance under a power cap and enables users and system administrators to understand how to best optimize application performance in these systems.

Keywords

Power Capping, Power Management, HPC

1. INTRODUCTION

Power consumption on capability-class supercomputers has become one of the key bottlenecks limiting overall system performance. This is motivated by the observation that the power used by each processor is no longer scaling downward with length scale due to thermal leakage, sometimes referred to as the end of Dennard (or MOSFET) scaling [7]. Projections indicate that if current power scaling trends are maintained, reaching the next milestone of a supercomputer with exaflop/s performance (10^{18} floating point operations per second) will require a power budget of several hundred megawatts [8]. This is not practical from a physical or economic standpoint, hence the HPC community's heightened interest in research aimed at mitigating the so called "power wall". Put differently, the community is expecting power usage to become an important limiting factor to achieving performance on capability-class systems.

One promising technique to limit power consumption is *power capping*. Power capping is a technique to control the peak power consumption of a system by monitoring current power draw and periodically selecting the highest performance hardware state while keeping the system within a fixed power constraint [16]. Power capping mechanisms are generally used with the goal of shifting the available power budget to where it is thought to be most beneficial to application performance. An underlying assumption of power capping is that due to power densities, powering on all hardware subsystems at full speed simultaneously will not be possible. Therefore, careful consideration must be made throughout the execution of an application as to which subsystems to turn on and which to let go "dark" (i.e., turn off), and similarly what proportion of the overall power budget should be given to each subsystem.

While power capping mechanisms have been available on individual hardware components for some time, for example RAPL [6] on Intel processors, the performance implications of this technique for HPC workloads running on capability-class HPC systems is not yet well understood. Therefore, in this work we investigate the performance of power capping on a number of key HPC workloads. Utilizing the power capping facilities of a next-generation capability-class system, this work demonstrates the following contributions:

- We present our early experiences using the Cray XC40's per-node power capping mechanism on a number of MPI benchmarks and a real-world application.
- We demonstrate that the independent, per-node power capping mechanism used on this platform can lead to poor scalability for certain MPI workloads, providing analysis of why this is the case – unsynchronized performance variability across nodes.
- We demonstrate that a simple technique – manually selecting a lower P-state setting on all nodes – can eliminate the poor scalability and improve application performance under a power cap by up to 22.7% in our experiments.
- Based on our experience, we suggest additional platform capabilities, such as the ability to dynamically change P-states during application execution and to

set CPU frequencies within groups rather than independently, that would be beneficial for enabling applications to stay within a power cap.

To the best of our knowledge, this is the first study to explore power capping in the context of a large-scale HPC architecture. While only one platform is examined, we expect the lessons learned in this study will also be applicable to other large-scale platforms incorporating similar power capping facilities.

The remainder of this paper is organized as follows: Section 2 describes the power management capabilities of the Cray XC40 architecture. We then describe our approach for evaluating the XC40's power capping capability in Section 3. Experimental results are presented in Section 4, along with a deeper analysis of the poor scalability that was observed for some cases. The implications of our results and our advice based on lessons learned are discussed in Section 5. Related work is discussed in Section 6, followed by conclusions in Section 7.

2. CRAY XC40 POWER MANAGEMENT

Our power capping experiments are performed on a 100 node Cray XC40 system at Sandia National Laboratories called Mutrino. Mutrino is an Application Readiness Testbed (ART) for the upcoming Trinity platform, which will consist of over 19,000 compute nodes. Mutrino is in effect a mini-Trinity that has all of the functional components of Trinity – I/O nodes, compute nodes, burst buffer nodes, water cooling infrastructure, etc. – but at a smaller scale. Both Trinity and Mutrino use identical "Haswell" compute nodes with the configuration shown in Figure 1b.

Figure 1 provides a high-level overview of the Cray XC40 power management architecture on Mutrino [5]. There is one system management workstation (SMW), shown in the upper left of Figure 1a, that system administrators use to boot and manage the system during operation. Users are not allowed access to the SMW. Users login to a set of front-end service nodes, shown at the bottom of Figure 1a, where they can compile their applications and launch applications onto the compute nodes. Users allocate compute nodes for their applications by making job allocation requests to the workload manager, shown in the upper right of Figure 1a. The workload manager allocates an exclusive set of compute nodes to each job request. If resources are not immediately available, the request is queued and satisfied once resources become available, subject to a scheduling policy.

The SMW is connected to all compute nodes via an Ethernet-based service network that is used for system monitoring and control. This network can be thought of as the "out-of-band" system control network, and is separate from the "in-band" high-performance Aries network that links all compute nodes together. Each XC40 compute node is instrumented with a power sensor that samples the node's power usage at 10 Hz. There is one "out-of-band" service processor (i.e., similar to a Baseboard Management Controller in commodity servers) for every four compute nodes that collects the 10 Hz power samples and once per second relays power usage information to an SQL database running on the SMW. This database can be queried by system administrators or the workload manager to get information about how power is being used in the system. Users are currently not allowed access to the SMW power database. User applications can, however, get "in-band" power and energy information by reading local files on each compute node (e.g., `/sys/cray/pm_counters/energy` contains the current energy counter, which accumulates the total Joules used by the node since boot).

The arrows in Figure 1a represent commands corresponding to the two primary power management capabilities of the Cray XC40 platform, 1) node-level power capping (in red), which is only available to system administrators, and 2) job-level static P-state control, which can be set by users at application launch time but cannot be changed afterwards. These two capabilities are described in more detail in the following sections.

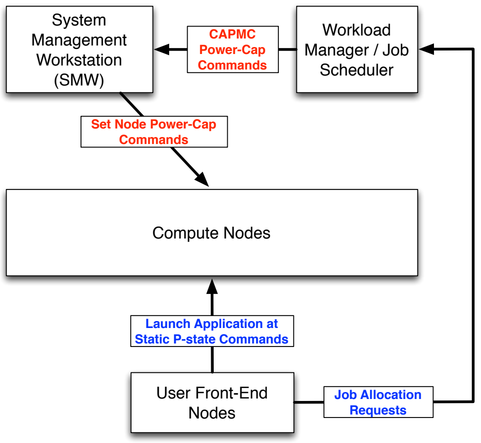
2.1 Node-level Power Capping

The Cray XC40 system provides a power capping mechanism that allows a power budget to be set for each compute node in the system. Firmware running on each compute node attempts to keep the node's power usage at or below this level for an unspecified sliding time window. If a node's power usage begins to exceed its power cap, the node is throttled to a lower performance level – e.g., by running at a lower P-state or performing clock gating – until the node's power usage falls below the power cap for an unspecified period of time. Node-level power capping in Mutrino is implemented using the Intel Node Manager firmware. Each Node Manager instance operates autonomously and independently with no cross-node coordination.

Activating a power cap is a privileged operation that requires access to the SMW. Cray provides a set of command line utilities on the SMW that allow system administrators to set up and activate power cap profiles on sets of compute nodes. Additionally, Cray developed a RESTful web API called CAPMC [19] that enables privileged system services, such as the workload manager, to perform power capping from remote (off-SMW) locations, as shown in the upper right of Figure 1b. Workload managers can use the CAPMC interface to get power usage information (e.g., how often the nodes in a job are being throttled) and then dynamically adapt power cap levels. Whether power capping functionality is accessed directly from the SMW or via CAPMC, activating and modifying power caps is a relatively expensive operation that requires several seconds to complete.

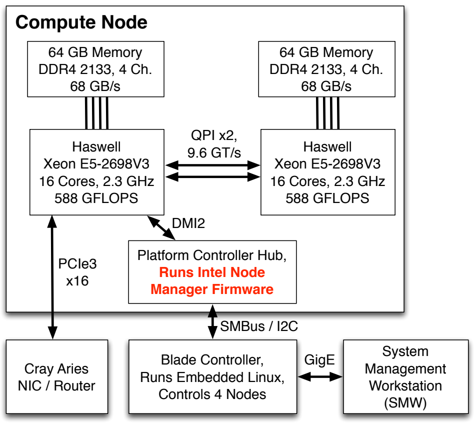
Internally the SMW power cap utilities send "Set Power-Cap" commands to each of the targeted compute nodes via the Ethernet monitoring and control network. As shown in Figure 1b, a power cap command will first arrive at the blade controller that manages the target node. Software running on the blade controller then relays the command to the target node's Platform Controller Hub (PCH). The PCH runs Intel's Node Manager firmware, which implements the power capping policy. Node manager continuously monitors the power usage of each Haswell processor and makes dynamic power management control decisions – such as changing CPU P-states – to maintain the desired power cap level. The exact details of how the node manager implements power capping are not disclosed by Intel but our empirical results presented in Section 4 provide some clues as to its operations.

Table 1 provides some example power cap profile configurations for compute node shown in Figure 1b. Cray names its power cap levels in terms of percentages, which represent the percentage within the range of a node's minimum and



System Management Workstation (SMW)
CAPMC Power-Cap Commands
Workload Manager / Job Scheduler
Set Node Power-Cap Commands
Compute Nodes
Launch Application at Static P-state Commands
User Front-End Nodes
Job Allocation Requests

(a) *System-Level*



Compute Node
64 GB Memory DDR4 2133, 4 Ch. 68 GB/s
64 GB Memory DDR4 2133, 4 Ch. 68 GB/s
Haswell Xeon E5-2698V3 16 Cores, 2.3 GHz 588 GFLOPS
QPI x2, 9.6 GT/s
Haswell Xeon E5-2698V3 16 Cores, 2.3 GHz 588 GFLOPS
DMI2
PCIe3 x16
Platform Controller Hub, Runs Intel Node Manager Firmware
SMBus / I2C
Cray Aries NIC / Router
Blade Controller, Runs Embedded Linux, Controls 4 Nodes
GigE
System Management Workstation (SMW)

(b) *Node-Level*

Figure 1: *Power Management Architecture of the Cray XC40 Platform*

maximum power usage. For this example compute node, the minimum power required to operate the node is 230 W and the maximum power is 415 W. Thus, a 50% power cap level represents a 322 W power cap in absolute terms. Cray chooses to represent power caps as percentages since they are somewhat easier to reason about than absolute wattages, particularly for systems with heterogeneous node types. Setting the power cap percentage level is the only configuration option that Cray exposes. In particular, it is not possible to change Intel Node Manager power capping configuration parameters such as the unspecified time duration used to calculate average power. This is a potential area for future investigation.

Table 1: *Example Power-Caps for node shown in Figure 1b*

Cray Power-Cap Setting	Power-Cap Per-Node (Watts)	Savings Potential (Watts)	Savings Potential (Percentage)
No Cap	$\sim$ 415 W	N/A	N/A
100%	415 W	$\sim$ 0 W	$\sim$ 0%
75%	369 W	46 W	11%
50%	322 W	93 W	22%
25%	276 W	139 W	33%
0%	230 W	185 W	45%

2.2 Job-level P-state Selection

Cray has chosen to expose static P-state selection to users at application launch time and to name their P-states directly with clock frequencies. If a user knows that their particular application will not benefit from running at the default "maximum performance" P-state setting, they can choose to manually select a lower P-state when they launch their application onto the compute nodes. The P-state selected is set for all cores within each node and this setting cannot be dynamically changed while the application is running.

Table 2 lists several of the available P-states for the compute node shown in Figure 1b, along with the percentage of peak performance delivered. The particular Intel processors used in this node operate at a maximum base frequency of 2.3 GHz and a minimum clock frequency of 1.2 GHz. At all P-states except for 2301000, the processors operate at the fixed frequency shown in the table. The 2301000 P-state enables Intel's "Turbo Boost" feature, which allows the processor's clock frequency to scale up from the 2.3 GHz base up to a maximum of 3.6 GHz, depending on factors such as the number of cores active and the currently available thermal headroom. An additional complication of the Intel Haswell processor is that heavy usage of AVX2 instructions causes the base frequency to be reduced somewhat [13], but this complication can be mostly ignored for the purposes of this paper.

Table 2: *Example P-states for node shown in Figure 1b*

Cray P-state Name	Clock Frequency (GHz)	Percent of Peak
2301000	2.3–3.6 (Turbo On)	> 100%
2300000	2.3	100%
2000000	2.0	87%
1900000	1.9	83%
1800000	1.8	78%
1600000	1.6	70%
1400000	1.4	61%
1200000	1.2	52%

3. APPROACH

This section describes the MPI workloads and testing procedure used to perform power capping experiments.

3.1 MPI Workloads

We evaluated two MPI benchmarks, High Performance Linpack (HPL) [1] and High Performance Conjugate Gradient (HPCG) [14], and one real MPI application, the S3D combustion code [4]. HPL and HPCG are from the Top500 [2] suite and represent different extremes in the spectrum of application behavior. HPL is highly compute bound, consisting of a dense LU factorization with $O(n^3)$ compute operations for $O(n^2)$ data movement. HPCG is highly memory bound, consisting primarily of low computational intensity operations like sparse matrix-vector products. The

S3D application performs numerical simulation of turbulent combustion using an explicit Runge-Kutta method. S3D is primarily network bound, with the dominant communication pattern being 3-D nearest-neighbor exchanges of ghost zones.

Test problems were configured for weak scaling from 1 to 96 nodes, with 32 MPI processes per node. Configurations for 1, 8, 32, 64, and 96 nodes were tested (32, 256, 1024, 2048, and 3072 MPI processes). The problem size for HPL was chosen to use about 24 GB of memory per node, scaled from N=56,000 for 1 node to N=549,000 for 96 nodes. HPCG was configured with the default 104x104x104 problem, using about 950 MB per MPI process (30 GB per node). To ensure that the same amount of work was done for all test configurations, HPCG was modified slightly to run for a fixed number of iterations rather than a fixed time period. S3D was configured for 48^3 gridpoints per node using an n-heptane/air chemical model with 52 transported species, 16 quasi-steady state species, and 283 chemical reactions. This configuration was chosen to be representative of the types of problems used in production S3D calculations. MPI topology mapping was performed to place a compact mini-box of the overall problem on each node, minimizing off-node communication.

HPL and HPCG were built using Cray compilers (`PrgEnv-cray/5.2.56`), since they were found to produce the best performance. S3D was built with GNU compilers (`PrgEnv-gnu/5.2.56`), as required. Optimization level -O3 was used in all cases.

3.2 Testing Procedure

Power capping experiments were performed while Mutrino was undergoing acceptance testing. Few users were allowed on the system so tests were done with no other jobs running. For each test window, a power cap setting was selected from Table 1 and installed on every compute node. Installing a power cap requires access to the SMW and root-level privilege. Once the power cap was active, typically within 10 seconds, each benchmark was executed at each of the p-state settings in Table 2 for each of the node counts tested, from 1 to 96 nodes. Tests with each of the power cap settings listed in Table 1 were performed, but results for the 75% setting are not presented because this configuration resulted in minimal performance impact.

For the HPL and HPCG tests, only one trial is plotted for each data point (power cap, p-state, and node count). Additional trials were run at the larger node counts but not shown as the variance is not visible in the figures. For S3D, five trials are plotted for each configuration, with error bars representing the minimum and maximum values displayed. The per-run problem wallclock time for HPL and HPCG varies from 5 to 30 minutes, dependent on node count, with the S3D times coming in a bit shorter at around a minute to 3 minutes per run.

All tests were performed with a 1-to-1 pinning of the 32 MPI processes per node to the 32 physical cores per node. Lastly, large pages were used for all tests via Cray's `craype-hugepages2M` module.

4. RESULTS

We first present the results of our scaling study for the Top 500 benchmarks, HPL and HPCG, followed by the scaling study for the full MPI application, S3D. Finally, we show MPI microbenchmarks and a view of the load imbalance induced by power capping. In all multi-part figures presented, one key is provided that is common between all of the subfigures within the figure.

4.1 Top500 Benchmarks

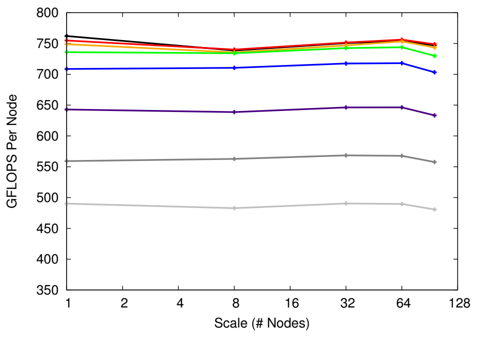
Figures 2 and 3 show the performance of HPL and HPCG, respectively, under different combinations of p-state and power cap configurations. Each of the subfigures within each figure shows results for a range of p-states from turbo mode down to 1.2 GHz under the same power cap. The number of nodes is scaled from a single node up to 96 nodes.

First consider the behavior of HPL. With no power cap imposed (Figure 2a), performance is similar for p-states at and above 1.9 GHz, likely due to the Haswell's automatic throttling of AVX2- heavy execution mixes, as mentioned in Section 2.2. Performance degrades at lower p-state settings, and varies little as node count increases. Imposing a cap at 50% of the allowed capping range (Figure 2b), reduces the performance at high p-states for all node scales, while performance at low p-states is not affected – their power usage never reaches the 322 W cap. In between the effect is pronounced: at a 1.8 GHz p-state, performance drops markedly as node count increases, dipping close to the 1.6 GHz p-state performance at 96 nodes. At a 25% / 276 W power cap (Figure 2c), the 1.6 GHz p-state results show a similar trajectory, underperforming even the 1.2 GHz p-state performance at 96 nodes. At the lowest allowed cap of 230 W (Figure 2d), the 1.4 GHz p-state results show a further decline.

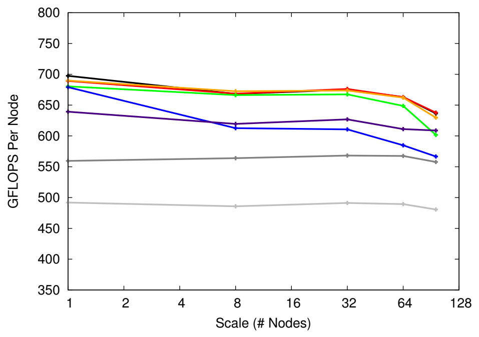
Turning to the results for HPCG, we observe that performance is relatively stable across p-states and node counts under no cap (Figure 3a) and a 50% / 322 W cap (Figure 3b). The similarity of the two graphs shows that regardless of p-state, HPCG's memory-intensive operations do not draw enough CPU power to reach a 322 W cap, unlike the more compute-intensive HPL. With a more restrictive 276 W / 25% cap (Figure 3c), HPCG performance does see an impact. In this case, performance is worst at 1.8 - 2.0 GHz p-states, only somewhat diminished at turbo mode and 2.3 GHz, and unimpacted by the 276 W cap at 1.2 - 1.6 GHz. At the most restrictive cap, 230 W (Figure 3d), the p-states of 1.2 - 1.6 GHz are now the worst performers. Moreover, the performance of these p-states falls off sharply with increased node count under this cap.

4.2 S3D Combustion Application

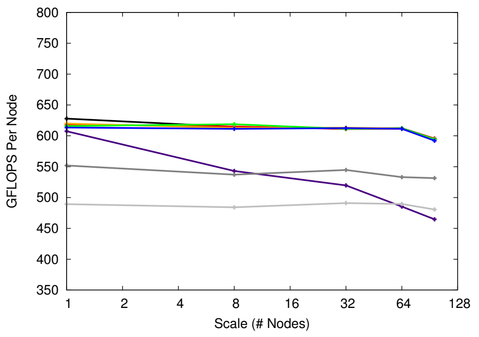
Section 4.1 showed that benchmark performance under a power cap varies depending on the combination of the wattage of the imposed power cap, the p-state frequency setting, and the benchmark characteristics. To investigate whether this behavior extends to a full real-world MPI application, we evaluate the performance of S3D under the same set of power cap and p-state configurations. The results are given in Figure 4, and as before, each subfigure shows performance under a different power cap. In the absence of a power cap (Figure 4a), turbo mode gives the best performance, with the other p-state settings showing steady performance decreases according to their relative frequencies. In all cases, there is a performance drop from 1 to 8 nodes as communication costs are introduced, and a continued, more gradual drop as node count increases further. With a 50% / 322 W cap (Figure 4b, running in turbo mode reaches the cap and the resulting application performance



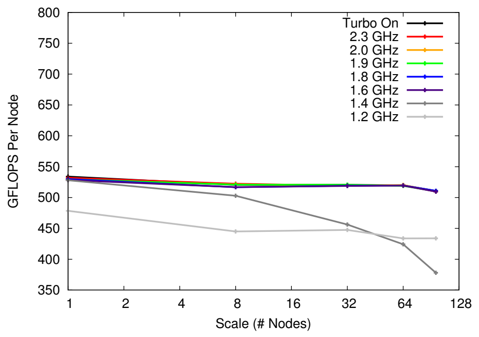
(a) ***HPL: No Cap (415 W)***



(b) ***HPL: 50% Cap (322 W)***

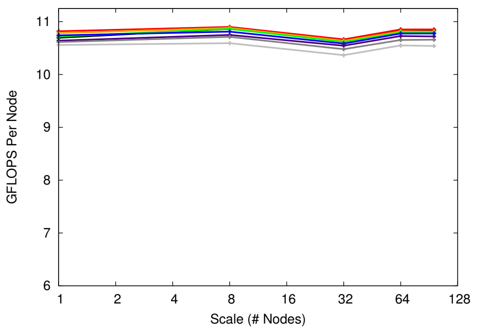


(c) ***HPL: 25% Cap (276 W)***

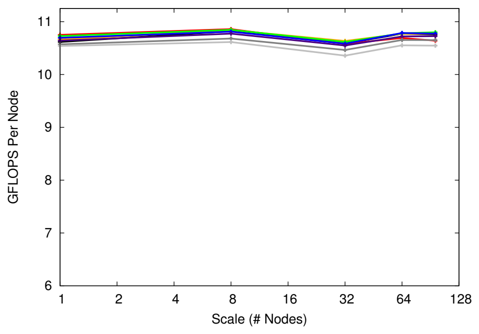


(d) ***HPL: 0% Cap (230 W)***

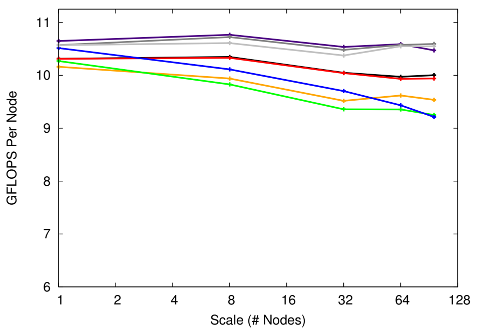
Figure 2: ***HPL Performance Scaling Under Power Capping***



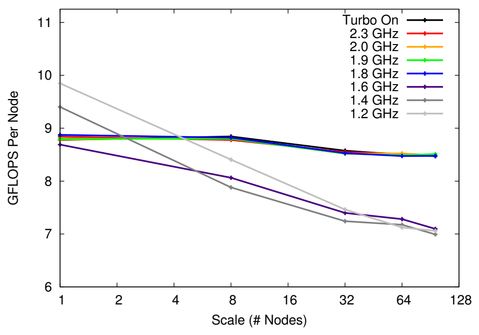
(a) ***HPCG: No Cap (415 W)***



(b) ***HPCG: 50% Cap (322 W)***



(c) ***HPCG: 25% Cap (276 W)***



(d) ***HPCG: 0% Cap (230 W)***

Figure 3: ***HPCG Performance Scaling Under Power Capping***

decreases to the same level as the 2.3 GHz p-state on the single node execution. Performance degradation worsens with node count – on 96 nodes, turbo underperforms the 1.8 GHz p-state. When the cap is tightened to 25% / 276 W (Figure 4c), turbo performance falls still lower and performance at the 2.3 GHz p-state drops below that of the 1.8 GHz p-state. Finally, under a 230 W cap (Figure 4d), performance of turbo, 2.3, and 2.0 GHz p-states degrades below 1.6 GHz performance, performance at the 1.9 Ghz p-state begins to drop, and the 1.2 - 1.6 GHz p-states maintain their same performance.

4.3 Measured Power Usage

Figure 5 shows the average power usage per node for the 96 node runs of each MPI workload. Average power is calculated for each run by dividing the total energy used, as reported by Cray's node-level energy counters, by wall clock runtime. For a given p-state, points below the "No Cap" value indicate that power capping is being activated to reduce power usage. Power capping may be occurring in other cases as well, for example if the application has bursty power usage behavior that isn't captured by the average power metric.

The results in Figure 5 can be correlated with Figures 2, 3, and 4 to see that when power capping is activated, performance is also usually reduced. For example, HPL uses approximately 355 W for the "No Cap" turbo case, as shown in Figure 5a. With a 50% (322 W) cap, HPL's power usage with turbo drops by 11% to 315 W while performance decreases by 14%. S3D's power usage with turbo begins to be curtailed at the 50% power cap setting, as shown in Figure 5c. With 50% cap and turbo p-state, S3D's power usage decreases by 10% while performance decreases 34% compared to "No Cap" turbo. This indicates that S3D is being more heavily impacted by power capping than HPL, and the general trend shown in Figure 4b is increasing impact with scale. HPCG's power usage, shown in Figure 5b, begins to be curtailed by power capping at the 50% level for turbo and at the 25% level for all other p-states. HPCG performance impact compared to "No Cap", shown in Figure 3, ranges from almost nothing with a 50% cap and turbo up to a 35% decrease with a 0% cap and 1.2 GHz p-state.

4.4 Analyzing Application Impact

In this section, we investigate characteristics of our tested workloads in order to better understand the performance impact of power capping displayed in the previous section. In this analysis we utilize MPI microbenchmarks and the CPU frequency values sampled during application execution. The former focuses attention on the interaction of MPI operations and power capping, demonstrating where manifest in application execution, in this case in collective operations. The latter gives a close-up view of the underlying system behavior that manifests in application load imbalance.

MPI Microbenchmark Performance.

In previous work [11, 10], we explored the break-down in performance for our example workloads – demonstrating how HPL's and HPCG's performance is sensitive to the performance of `MPI_Allreduce()`. S3D, on the other hand, performs very few Allreduce operations, or MPI collectives in general, and displays sensitivity to point-to-point performance. Therefore, we look to MPI latency, bandwidth, and `MPI_Allreduce()` microbenchmark performance to help explain the observed performance impacts.

Figure 6 shows the performance of MPI point-to-point messaging for ping-pong one-way latency and bandwidth with no power cap set. Ping-pong performance with 0%, 25%, 50%, and 75% power caps active was virtually identical the no cap results, so plots are omitted due to space. Small message latency, shown in Figure 6a varies from 1.4 μs with turbo to 2.4 μs using lowest available p-state, 1.2 GHz. Most p-states show low latency variability except for 2.3 GHz, which has noticeable variation for all message sizes. Figure 6b shows that the bandwidth difference between turbo and 1.2 GHz p-states varies from 41% for small messages to 0% for large messages, where all cases reach the asymptotic bandwidth.

Figure 7 shows the performance for 8-byte MPI Allreduce (The most common size Allreduce operation found in HPCG and HPL). With no power cap active (Figure 7a), performance scales regularly and there is little variability. Unlike with ping-pong, allreduce performance is impacted by power capping. With a 50% power cap active (Figure 7b), allreduce performance with turbo is increasingly impacted with scale, becoming up to 59% worse than when using the 2.0 GHz p-state at 96 nodes (10% worse on average). This trend is amplified with 25% and 0% power caps (Figures 7c and 7d), where the performance of the turbo and 2.3 GHz p-states becomes extremely noisy. The performance of the lower p-states, 2.0, 1.6, and 1.2 GHz are not significantly impacted, presumably because they result in a power level that does not trigger the power capping mechanism.

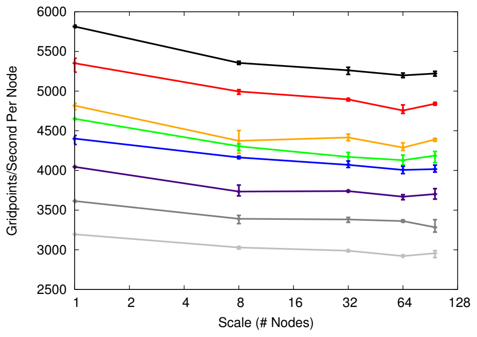
The Allreduce results do help explain the performance impacts we see with HPL and HPCG. As power cap decreases, Allreduce performance variability increases, leading to application slowdowns. The bandwidth and latency numbers, however, do not explain the performance of S3D. To better explain its results, we took a closer look at the underlying system behavior, specifically the CPU per-node frequency data.

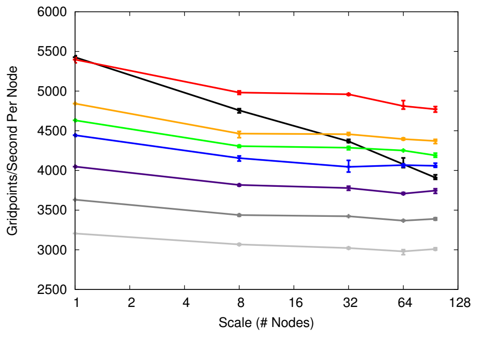
Per-node Frequency Settings.

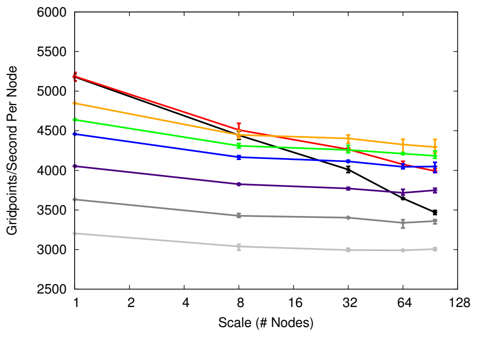
The communication pattern in S3D is nearest-neighbor. Like Allreduce, this pattern requires all nodes to finish their communication before the application can proceed (i.e., to the next time step in the simulation). In fact, previous work [11] has demonstrated that these nearest neighbor exchanges can have significant global slowdown even when a small subset of nodes experience performance slow-downs.

To dive deeper into the effects of power capping on S3D, we sampled the CPU frequency on each node of the machine at 10 Hz intervals. Figure 8 shows sampled data from two executions of S3D on 96 nodes under a 230 W power cap. In each graph, samples from ten of the nodes are shown, one row per node. Time proceeds from left to right across the x-axis. Each point is colored based on is observed operating frequency in Hz, as shown in the legend at right. Figure 8a shows that at p-state 1.8 GHz, a consistent CPU operating frequency of 1.8 GHz is maintained throughout the execution, as the power usage remains below the imposed power cap.

Figure 8b demonstrates the very different behavior resulting from execution in turbo mode. Execution begins with nodes running at high frequency. As power usage reaches the cap, the CPU frequency is throttled. Throughout execution,


(a) *S3D: No Cap (415 W)*


(b) *S3D: 50% Cap (322 W)*


(c) *S3D: 25% Cap (276 W)*

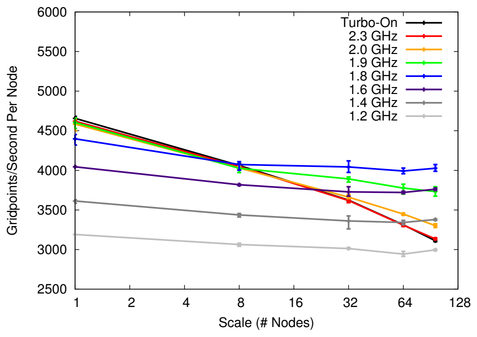
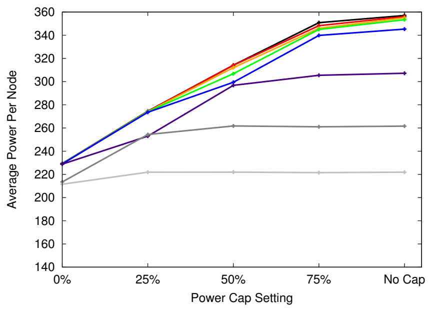
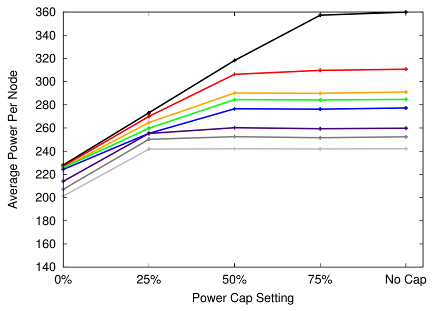

(d) *S3D: 0% Cap (230 W)*

Figure 4: ***S3D Performance Scaling Under Power Capping,*** 48^3 ***Problem***


(a) *HPL*


(b) *HPCG*

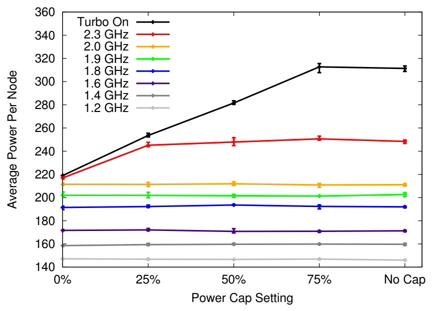
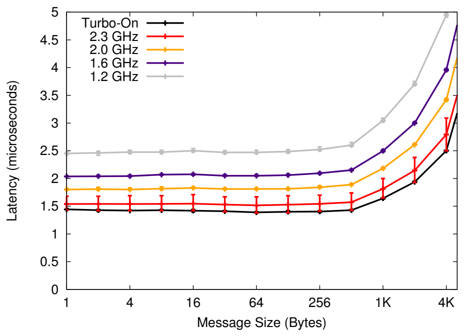

(c) *S3D*

Figure 5: ***Average Power Usage Per Node Under Power Capping for 96 Node Runs***


(a) *Small Message Latency*

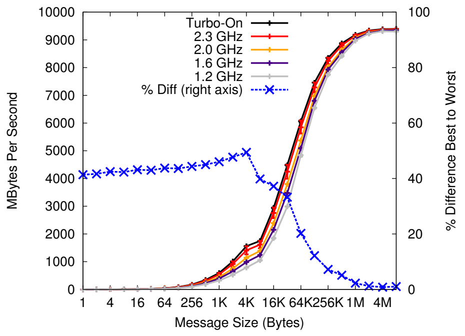
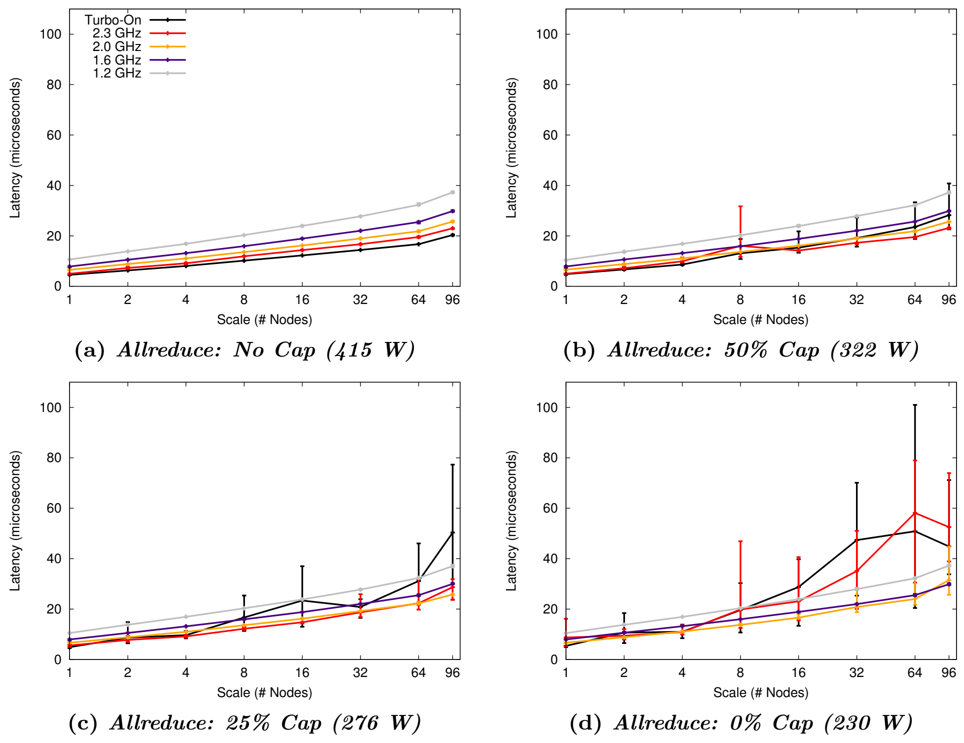

(b) *Bandwidth*

Figure 6: ***MPI PingPong Performance for No Cap (415 W) at Different P-states***



(a) ***Allreduce: No Cap (415 W)*** **(b)** ***Allreduce: 50% Cap (322 W)***

(c) ***Allreduce: 25% Cap (276 W)*** **(d)** ***Allreduce: 0% Cap (230 W)***

Figure 7: ***MPI 8-byte Allreduce Scaling Under Power Capping***

the frequencies vary quite widely, even down to 1.6 GHz. At a particular point in time, some nodes are operating at a high frequency while others are operating at lower frequencies. The result for a tightly-coupled MPI application like S3D is load imbalance, in which slower nodes hold back the progress of faster nodes. As with load imbalance due to OS noise demonstrated in previous work [11], this particularly inhibits performance at scale.

5. DISCUSSION

In this section we discuss the implications of the results presented in the previous section. First, we summarize the performance of our workloads under a power cap. We then outline the importance of avoiding the gap. Finally, we conclude the section with advice on mitigating performance impacts. In the former two cases, we describe what this means for applications designers, OS/runtime developers, and HPC vendors and system integrators.

Summary of Results.

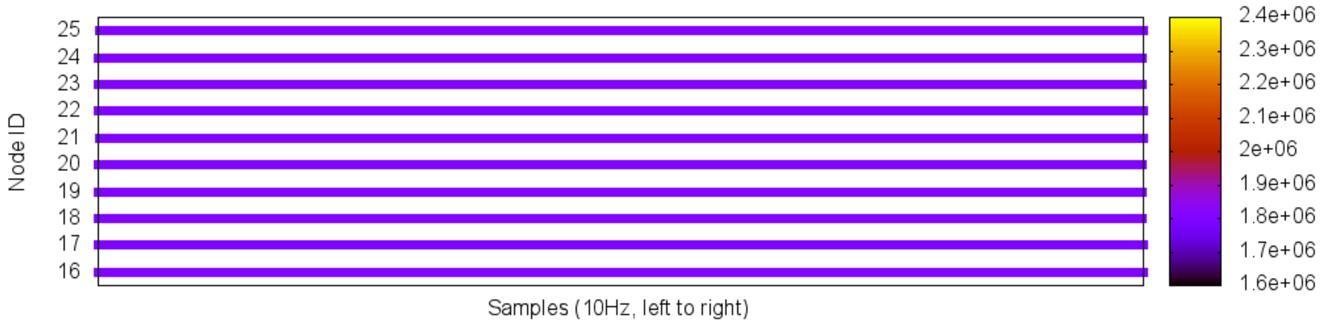
Based on the data presented in Section 4, Table 3 shows the optimal p-state under each power cap setting for HPL, HPCG, and S3D. While HPL performance does degrade under a power cap depending on both the power cap and the p-state, turbo mode consistently out-performs the other p-state settings. For HPCG, turbo mode is best in all but one case. Unlike either of the benchmarks, the S3D application requires lowering the p-state as the power cap becomes more restrictive in order to achieve the best performance possible under that cap. As shown in Section 4.4, power capping induces slow-downs on different nodes at different times, so for a tightly-coupled MPI application, running at a p-state that avoids reaching the cap is a profitable strategy to maximize performance.

Advice to Users and Implementers: Avoid the Cap.

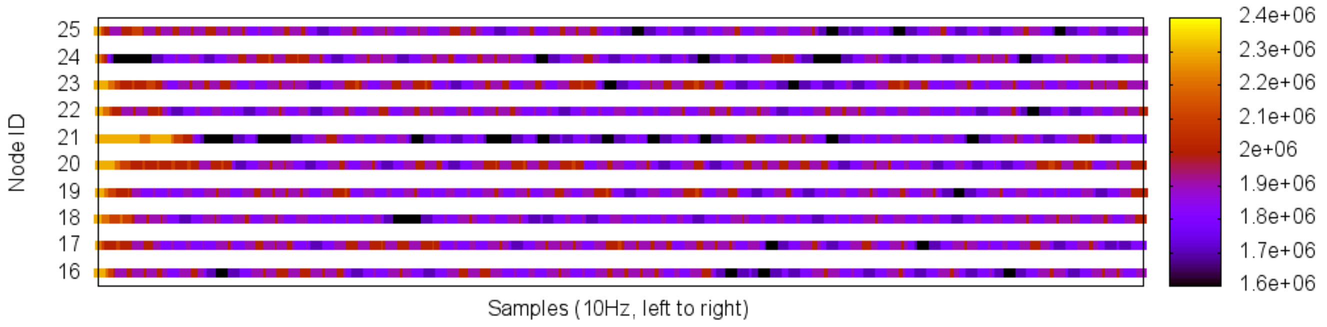
Overall, our work demonstrates the importance of avoiding hitting the power cap. With the rise of power capping on emerging HPC systems, application developers face the reality that their application performance will be severely curtailed if they are unable to ensure that their power draw does not exceed the cap. We have observed that adjusting the p-state is an effective mechanism to do this. Unfortunately, Cray XC40 power management only exposes a static p-state control. Since application power characteristics under a particular power cap are difficult to predict *a priori*, and the level of the imposed power cap may change during execution, system implementers should expose dynamic p-state control so that adjustments can be made to avoid the cap. To relieve the burden to the end user, given such a capability, the MPI run time, operating system, or another system software component could be enhanced to adaptively adjust the p-state during execution to remain just below the power cap.

Mitigating Impacts: Coordinated CPU Throttling.

For scenarios where avoiding the cap is not possible, mitigating the impacts of the cap may make sense. One possible source of inspiration for mitigation methods is from an analogy with OS noise [21, 10, 15]. Though studied for over 20 years, recent work has shown that increasing the synchronization of the source of the noise can lead to benefits in performance [11]. It is important to note that total synchronization is not necessary needed. Typically what is needed is



(a) ***P-state at 1.8 Ghz***



(b) ***Turbo***

Figure 8: ***Observed CPU Frequency over Time: S3D Under 230 W Power Cap,*** 48^3 ***Problem***

loose synchronization with processing elements that communicate frequency. In the context of power caps, this means adjusting CPU frequencies with nodes that communicate frequently, rather than the current independent approach. This functionality requires both support at a hardware-level to specify power caps with groups of nodes and a runtime capable of determining the optimal groups.

Table 3: ***Best P-state/P-cap Setting for 96 Node Runs***

Power-Cap Setting	HPL	HPCG	S3D
No Cap	Turbo	Turbo	Turbo
75%	Turbo	Turbo	Turbo
50%	Turbo	Turbo	2.3 (18.1%)
25%	Turbo	1.4 (6%)	2.0 (19.2%)
0%	Turbo	Turbo	1.8 (22.7%)

6. RELATED WORK

Power capping has long been a topic of considerable interest in the commercial data center and server space. Fan et al. examine theoretical potential of power capping and provisioning for average power usage in large-scale datacenters, e.g. at Google [9]. Lefurgy et al. implement power capping in a blade server based on control theory methods.[16]. Lo et al. implement a system to limit power to the minimum amount required to maintain search response times within service level agreement terms [17].

Studies on power management for high-performance scientific computing have recently been accelerated both by the recognition that power will be a key constraint in future HPC systems and by the availability of the Running Average Power Limit (RAPL) [6] feature for CPU-level power limits in Intel SandyBridge processors. Rountree et al. show that part-to-part variability in power efficienty characteristics result in performance variability under a RAPL power cap [23]. Patki et al. propose overprovisioning systems with respect to power and allocating it based on application characteristics rather than worst-case assumptions [20]. Sarood et al. demonstrate that adding additional low power nodes to an execution may be improve performance compared to running fewer high power nodes [24]. Porterfield et al. boost power on processors that fall behind due to hardware performance variability under power cap. [22]. Several power-aware schedulers for HPC have been proposed [3, 26, 18]. Related efforts to understand application and MPI implementation power characteristics include a single-node power capping experiment of a magnetohydrodynamics application [12] and a study of energy usage of MPI primitives on four nodes [25].

The studies listed above have used either simulation or 1-64 nodes of SandyBridge or earlier processors, implementing power capping through direct manipulation of RAPL. In contrast, our study uses Intel's newer Haswell processors, designed for more advanced power management, with power capping applied through Cray's production XC40 power management infrastructure to show MPI application impact at up to 96-node scale for a wide range of p-state and power cap combinations.

7. CONCLUSION

Our early evaluation of XC40 system power capping impact on MPI application performance is a key component of our preparations for power management on our upcoming 30 petaflop Trinity supercomputer. The comprehensive testing of power cap and p-state combinations for several workloads shows considerable performance degradation due to load imbalance when nodes are throttled by the system as they reach the power cap. To optimize performance in the presence of power capping, we observe that an application should avoid reaching the power cap. Adjusting the p-state is a mechanism to achieve this avoidance, and thus we recommend that vendors expose dynamic p-state controls for use by end users or system software.

8. ACKNOWLEDGMENTS

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.

References

[1] HPL. `http://www.netlib.org/benchmark/hpl/`.

[2] List of TOP500 supercomputer sites. `http://www.top500.org/`.

[3] D. Bodas, J. Song, M. Rajappa, and A. Hoffman. Simple power-aware scheduler to limit power consumption by HPC system within a budget. In *Proc. 2nd International Workshop on Energy Efficient Supercomputing*, E2SC '14, pages 21–30, 2014.

[4] J. H. Chen, A. Choudhary, B. d. Supinski, M. DeVries, E. R. Hawkes, S. Klasky, W. K. Liao, K. L. Ma, J. Mellor-Crummey, N. Podhorszki, R. Sankaran, S. Shende, and C. S. Yoo. Terascale direct numerical simulations of turbulent combustion using S3D. *Computational Science & Discovery*, 2, 2009.

[5] Cray Inc. *Monitoring and Managing Power Consumption on the Cray XC System*, April 2015. http://docs.cray.com/books/S-0043-7203/S-0043-7203.pdf.

[6] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le. RAPL: Memory power estimation and capping. In *Proc. International Symposium on Low-Power Electronics and Design (ISLPED)*, Aug 2010.

[7] R. H. Dennard, F. H. Gaensslen, H. nien Yu, V. L. Rideout, E. Bassous, Andre, and R. Leblanc. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE J. Solid-State Circuits*, page 256, 1974.

[8] R. L. et al. Top ten exascale research challenges. Technical report, U.S. Deparment of Energy, Office of Science, Office of Advanced Scientific Computing Research, Washingtin D.C., 2014.

[9] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *Proc. 34th Annual International Symposium on Computer Architecture*, ISCA '07, pages 13–23, 2007.

[10] K. B. Ferreira, P. Bridges, and R. Brightwell. Characterizing application sensitivity to OS interference using kernel-level noise injection. In *Proc. 21st International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '08, 2008.

[11] K. B. Ferreira, P. Widener, S. Levy, D. Arnold, and T. Hoefler. Understanding the effects of communication and coordination on checkpointing at scale. In *Proc. 27th International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '14, pages 883–894, Nov. 2014.

[12] K. Fukazawa, M. Ueda, M. Aoyagi, T. Tsuhata, K. Yoshida, A. Uehara, M. Kuze, Y. Inadomi, and K. Inoue. Power consumption evaluation of an MHD simulation with CPU power capping. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pages 612–617, May 2014.

[13] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. An energy efficiency feature survey of the Intel Haswell processor. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, May 2015.

[14] M. A. Heroux and J. Dongarra. Toward a new metric for ranking high performance computing systems. Technical Report SAND2013-4744, Sandia National Laboratories, Albuquerque, NM, 2013.

[15] T. Hoefler, T. Schneider, and A. Lumsdaine. Characterizing the influence of system noise on large-scale applications by simulation. In *Proc. 23rd International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '10, 2010.

[16] C. Lefurgy, X. Wang, and M. Ware. Power capping: a prelude to power shifting. *Cluster Computing*, 11(2):183–195, 2008.

[17] D. Lo, L. Cheng, R. Govindaraju, L. A. Barroso, and C. Kozyrakis. Towards energy proportionality for large-scale latency-critical workloads. In *Proc. 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pages 301–312, 2014.

[18] A. Marathe, P. E. Bailey, D. K. Lowenthal, B. Roundtree, M. Schulz, and B. R. de Supinski. A run-time system for power-constrained HPC applications. In *Proc. ISC High Performance Conference*, ISC '15, July 2015. To appear.

[19] S. J. Martin, D. Rush, and M. Kappel. Cray advanced platform monitoring and control (CAPMC). In *Proc. Cray Users' Group Technical Conference (CUG)*, 2015.

[20] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski. Exploring hardware overprovisioning in power-constrained, high performance computing. In *Proc. 27th ACM International Conference on Supercomputing*, ICS '13, pages 173–182, June 2013.

[21] F. Petrini, D. J. Kerbyson, and S. Pakin. The case of the missing supercomputer performance: Achieving optimal performance on the 8,192 processors of ASCI Q.

In *Proc. 16th International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '03, 2003.

[22] A. Porterfield, R. Fowler, S. Bhalachandra, B. Rountree, D. Deb, R. Lewis, and B. Blanton. Application runtime variability and power optimization for exascale computers. In *International Workshop on Runtime and Operating Systems for Supercomputers*, ROSS '15, June 2015. To appear.

[23] B. Rountree, D. H. Ahn, B. R. de Supinski, D. K. Lowenthal, and M. Schulz. Beyond DVFS: A first look at performance under a hardware-enforced power bound. In *IEEE 26th International Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW)*, May 2012.

[24] O. Sarood, A. Langer, L. Kale, B. Rountree, and B. R. de Supinski. Optimizing power allocation to CPU and memory subsystems in overprovisioned HPC systems. In *2013 IEEE International Conference on Cluster Computing*, Cluster '13, Sept 2013.

[25] A. Venkatesh, K. Kandalla, and D. Panda. Evaluation of energy characteristics of MPI communication primitives with RAPL. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 938–945, May 2013.

[26] Z. Zhang, M. Lang, S. Pakin, and S. Fu. Trapped capacity: Scheduling under a power cap to maximize machine-room throughput. In *Energy Efficient Supercomputing Workshop (E2SC), 2014*, pages 41–50, Nov 2014.