Introducing Application Awareness Into a Unified Power Management Stack

Daniel C. Wilson*‡, Siddhartha Jana*, Aniruddha Marathe†, Stephanie Brink†, Christopher M. Cantalupo*,
Diana R. Guttman*, Brad Geltz*, Lowren H. Lawson*, Asma H. Al-rawi*, Ali Mohammad*,
Fuat Keceli*, Federico Ardanaz*, Jonathan M. Eastep*, Ayse K. Coskun‡
*Intel Corporation, †Lawrence Livermore National Laboratory,‡Boston University
Email: *{siddhartha.jana, christopher.m.cantalupo, diana.r.guttman, brad.geltz, lowren.h.lawson, asma.h.al-rawi,
ali.mohammad, fuat.keceli, federico.ardanaz, jonathan.m.eastep}@intel.com,
†{marathe1, brink2}@llnl.gov, ‡{danielcw, acoskun}@bu.edu,

***Abstract*—Effective power management in a data center is critical to ensure that power delivery constraints are met while maximizing the performance of users' workloads. Power limiting is needed in order to respond to greater-than-expected power demand. HPC sites have generally tackled this by adopting one of two approaches: (1) a system-level power management approach that is aware of the facility or site-level power requirements, but is agnostic to the application demands; OR (2) a job-level power management solution that is aware of the application design patterns and requirements, but is agnostic to the site-level power constraints. Simultaneously incorporating solutions from both domains often leads to conflicts in power management mechanisms. This, in turn, affects system stability and leads to irreproducibility of performance. To avoid this irreproducibility, HPC sites have to choose between one of the two approaches, thereby leading to missed opportunities for efficiency gains.**

This paper demonstrates the need for the HPC community to collaborate towards seamless integration of system-aware and application-aware power management approaches. This is achieved by proposing a new dynamic policy that inherits the benefits of both approaches from tight integration of a resource manager and a performance-aware job runtime environment. An empirical comparison of this integrated management approach against state-of-the-art solutions exposes the benefits of investing in end-to-end solutions to optimize for system-wide performance or efficiency objectives. With our proposed system–application integrated policy, we observed up to 7% reduction in system time dedicated to jobs and up to 11% savings in compute energy, compared to a baseline that is agnostic to system power and application design constraints.

***Index Terms*—Energy-aware systems, Scheduling, Information resource management**

I. Introduction

Sizing a data center's power supply involves a trade-off between peak performance of individual workloads, and the total number of hosts available to run those workloads. Invest too little in power, and you run the risk that a cluster will perform poorly because it needs more power than can be delivered. But if you invest too much in power delivery equipment, you sacrifice capital and floor space that could have gone to acquiring more computing infrastructure. Several examples involving under-utilization of procured power exist in the real world. For example, Figure 1 shows the average power draw of the Quartz system at LLNL over a period of one year. The system is rated to consume 1.35 MW as indicated by the dashed line, but the average power draw remains at 830 kW. Another example is the Cori system at NERSC which is rated to consume 5.7 MW, but its peak power has only reached 4.6 MW, and it typically only consumes 3.9 MW [1].

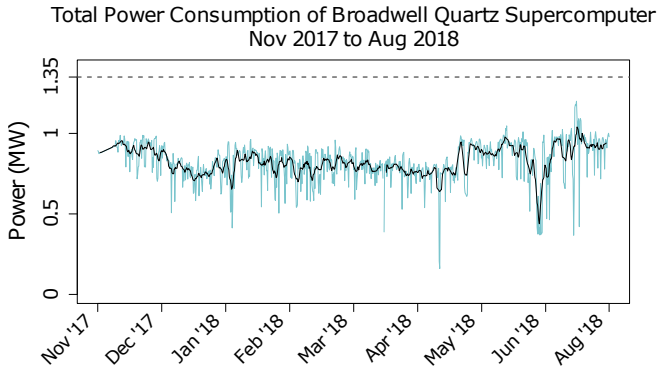

Fig. 1. Power usage of Quartz system at LLNL over a period of one year. The dashed line shows the peak power rating of the system. The solid blue line shows actual instantaneous power usage whereas the solid black line shows moving average of the instantaneous power draw over a window of one day.

Power delivery infrastructure must ensure that a site's total power consumption does not exceed the deliverable power capacity. The site's power draw depends on non-uniform demands of its compute nodes, servicing different applications. State-of-the-art approaches for power management of the compute nodes can be divided into two categories: (1) system-level control based on the *observed* consumption (e.g., [2], [3]), and (2) application-level control based on the *required* power (e.g., [4]). Solution-(1) has visibility into the site-level constraints, but is agnostic to the actual application demands. Solution-(2) is aware of the application design requirements, but remains oblivious to the available power at the system level. This paper presents an opportunity analysis of approaches that leverage the best of both solutions by integrating system-level and application-level power management policies.

The **key takeaways** from this paper are as follows:

- A siloed power management strategy, isolated to a single layer of the system stack, does not sufficiently optimize system performance or efficiency objectives. Moreover, having multiple such solutions run simultaneously without coordination leads to unpredictability. For example, if power limits are controlled through the same hardware interface by both a resource manager and a job runtime environment, one layer may unintentionally overwrite limits set by the other layer.
- The solution is for HPC vendors to use holistic approaches that apply power management techniques that run in tandem to address both system-level and application-level power requirements.
- Power delivery infrastructure can handle more aggressive power limits with less impact to quality of service by coordinating between layers. Such coordination can ultimately increase the amount of *science per watt*.

The paper presents an opportunity analysis for introducing *interoperability* among multiple layers of the system software stack. The **novel contributions** of this paper are as follows:

1) We propose a workload-aware and system-power-aware policy that reduces time to solution in a power-limited cluster by distributing power both across and within workloads in a performance-aware manner.
2) We identify a set of application design characteristics that emulate different power and performance trade-offs seen in HPC applications, and we design a microbenchmark[1] to trigger these different conditions.
3) We evaluate the proposed policy against two other dynamic policies with either system power awareness or workload awareness, and an additional policy with neither feature. Our evaluations at scale show up to 7% system time reduction and up to 11% CPU energy savings, versus the policy that has neither feature.

This paper is organized as follows. We first motivate the implementation of a multi-level system-wide power management stack. Next, we describe the power management policies, along with the synthetic benchmark and cluster environment used for evaluation. Then we describe the matrix of workload combinations and power cap levels. We explain our results before moving on to discuss related approaches to system-wide power management. We conclude with takeaways and the next steps that should follow this work.

II. Motivation

HPC sites have different power and energy requirements, depending on their environment and the types of workloads they run. The Energy Efficient High Performance Computing Working Group (EEHPC-WG) performed a survey of Top500 sites working with energy- and power-aware job scheduling and resource management [5], showing a need for awareness of dynamic factors in both power supply and demand at a site.

[1]The synthetic kernel used in this study is hosted on a public repository at https://github.com/dannosliwcd/arithmetic-intensity.git

Within a site, a system can be composed of many nodes that react differently under the same workloads. Even with a homogeneous system containing all nodes with the same configuration, hardware variation can have a significant impact on workload efficiency and performance [6].

Workload characteristics also have a significant impact on performance and efficiency [7]. For example, workloads often have compute-intensive and non-compute-intensive phases. Non-compute-intensive phases could result from pipeline stalls due to dependencies on memory, network resources, storage devices, accelerators, thread scheduling, etc. Since CPU activity is a major contributor to total system power, and can be controlled with low-latency interfaces, this paper studies the impact of controlling CPU power to meet total system power and performance constraints.

Even *within* a phase of an application, workloads can require varied demand for compute resources across processes, called *workload imbalance* in this paper. Workload imbalance in bulk-synchronous workloads can make the application's overall performance largely insensitive to the performance of some of its parts, since only the process in the workload's critical path will have an immediate impact on aggregate performance. Several efforts have identified software critical paths for optimization in development and for modifying an execution environment to exploit that insensitivity [4], [8].

The **theme among our motivations** is that some performance and efficiency factors are most visible with a system-wide perspective, and others are more visible within the scale of a single workload execution. In this paper, we evaluate power management policies with varying degrees of visibility into system constraints and workload properties. Through our evaluation, we demonstrate an opportunity for the HPC community to work toward standardization of power management policies across layers of a system stack. This is in alignment with the charter of the HPC PowerStack consortium [9].

III. System-Level Power Management Policies

To evaluate the potential for performance improvement, we design a new power management policy, `MixedAdaptive`. We also implement baseline power management policies for comparison purposes.

A. Performance-Aware System Power Management

The proposed `MixedAdaptive` policy enables a resource manager to share power across jobs in a power-aware manner. This policy's power awareness is made available to the resource manager by a job runtime, which can search at execution time for the distribution of available power that minimizes elapsed time per iteration in a workload. Our experiments utilize the GEOPM [4] job runtime to apply energy- and performance-aware power management algorithms. GEOPM includes multiple plugins (called *agents*) for monitoring and managing efficiency.

While existing job runtimes are able to utilize performance awareness, they are not currently able to establish an execution-time feedback loop with a resource manager. We

emulate such a feedback loop in our experiments by first running our workloads under the GEOPM *power_balancer* agent and identifying the steady-state power consumption for each workload host. The *power_balancer* agent reduces the power limit where it does not impact performance, and redistributes that power where it can improve performance, all during execution. Our experiments use the final power distribution from a pre-characterization run with that agent to select how much power each job is allocated by a power-limited resource manager, and when determining how a job runtime would distribute that allocated power within each job.

The **`MixedAdaptive`** power distribution steps are as follows:

1) Uniformly distribute the system power limit among hosts across all jobs.
2) Decrease the allocated power of each host down to the amount of power needed on that host, as determined by the previously described *power_balancer* pre-characterization runs. The total amount of decreased power is now considered deallocated. If there is a significant enough power shortage, the surplus can be as low as zero watts.
3) Uniformly distribute the deallocated power among hosts that need more power to meet their characterized performance, at most up to the characterized power. Repeat this step until no deallocated power remains, or all hosts have been assigned their needed power.
4) If there is a power surplus, allocate the remainder of power across all hosts with a weighted distribution. The weight of each host is determined by the distance from the host's minimum settable power limit to the host's allocated power from previous steps.

B. Baseline Power Management Policies

In addition to the previously-described *power_balancer* characterization, our baselines also use workload characterization data from the GEOPM *monitor* agent, which simply reports requested metrics of interest, such as energy and time, without modifying system behavior.

Our methods for determining policy power allocations based on GEOPM characterization data are explained below.

For the **`Precharacterized`** policy, a user *pre-characterizes* a workload, and submits the job with a cap equal to the average power consumption at the most power-hungry node. This policy does not consider system-wide power limits.

For the **`StaticCaps`** policy, system power is uniformly distributed to all nodes in the cluster. A *static cap* is applied for each job, using the max of average powers from all nodes in the job's *monitor* characterization run. Note that this policy's final state is the same as the initial state of the `MinimizeWaste` and `MixedAdaptive` power-sharing policies.

`MinimizeWaste` shares system power across hosts, to minimize unused power budget. This policy is intended to statically emulate the *dynamic* approach documented in SLURM's real-time power management feature [3], which *is full-system-aware*. Our policy first distributes power caps across jobs. It then reduces the budget for low-power jobs to minimize unused (wasted) power budgets, and evenly redistributes power to high-power jobs. The power is removed from and added to jobs based on the observed *performance-agnostic* power usage (obtained from GEOPM reports) for each workload. Surplus power is redistributed, weighted by the difference between minimum settable power and currently assigned power.

For the **`JobAdaptive`** policy, system power is *dynamically* shared within jobs to maximize performance, but power cannot be shared across different jobs. In other words, the policy is *not full-system-aware*. The system power cap is initially distributed uniformly across jobs. Power is further distributed among hosts within each job, based on the *performance-aware* characterization data (from GEOPM reports). If any of the nodes are assigned a power limit that exceeds an evenly-distributed power cap, then all nodes in the job have their power caps reduced by the percentage of their current power consumption that corrects that violation.

IV. Workload Design

We use a synthetic kernel to provide fine-grained control of factors that dictate the energy/power signatures of a target platform. This section gives an overview of the different application characteristics that are captured by the kernel, followed by results of a characterization study identifies energy/power bounds of the kernel. We also present a roofline analysis of the benchmark to show that the kernel exhibits the expected range of memory- and compute-boundedness.

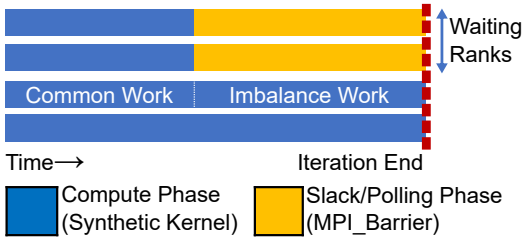

Fig. 2. Design characteristics of the synthetic microbenchmark used for emulating typical HPC application properties.

A. Compute Intensity Benchmark

Our synthetic kernel is derived from a benchmark used by Choi et al. to evaluate a roofline model of energy [10]. The makeup of the kernel is illustrated in Figure 2. The typical application characteristics that impact the energy/power signature of an application are as follows:

- Computational intensity: This is the ratio of compute work to memory work, in FLOPs/byte. Intensity directly impacts the maximum possible CPU throughput [11].
- Vector length of instructions: A fixed computational intensity can have power-performance trade-offs that vary with the length of vector instructions. We use vector fused

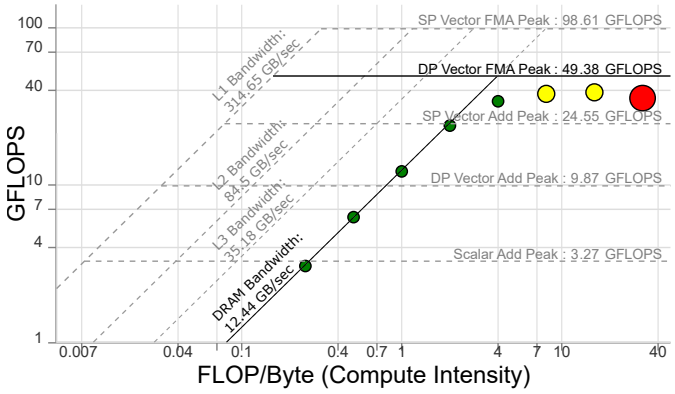

Fig. 3. The roofline plot of the synthetic kernel when executed on the target platform. The X-axis corresponds to the computational intensity (FLOPs/Byte). The Y-axis corresponds to the achieved computational throughput (in GFLOPS). The colored dots correspond to the runs with the synthetic kernel. The bold continuous black line corresponds to the maximum achievable performance on the target platform, given our workload configurations.

multiply-add (FMA) and load instructions to capture a high ratio of instructions per cycle.

- Excess time on a non-critical path: Processes in large-scale bulk-synchronous and fork-join applications frequently converge at synchronizing points within the application. Such applications can end up with some processes polling at synchronizing points while making no application progress, but still consuming energy. Our microbenchmark captures such energy sinks by controlling the work performed on the non-critical path.
- Percent of waiting ranks: This is the fraction of synchronizing processes on the non-critical path. A higher percentage corresponds to a higher number of processes polling at a barrier and consuming energy without making any application progress.

We verify that the kernel covers the full spectrum of achievable throughput of the platform by overlaying the kernel's performance on top of the system's roofline plot. Figure 3 shows that the kernel's performance at each configuration reaches the peak performance (in GFLOPS) of the system, bounded by the DRAM bandwidth and double-precision floating point vector fused multiply-add operations. The data points corresponding to the kernel behavior are depicted by colored dots in the graph, which is generated by the Intel Advisor tool [12].

B. Workload Characterization

For our experiments, we need to know: (a) the maximum power each workload consumes under no power constraints, and (b) the minimum power each workload needs to complete execution. Greater differences between these metrics indicate more opportunity for leveraging application awareness in power-constrained scenarios.

We obtain Metric-(a) by executing each workload with the GEOPM *monitor* agent across 100 test nodes. The heat map in Figure 4 summarizes our measurements.

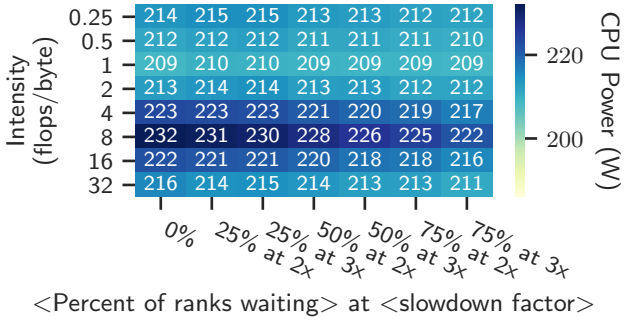

Fig. 4. Total CPU power per node for different workload configurations, when running the benchmark variant that uses 256-bit (ymm) vector registers with no power limit under the GEOPM *monitor* agent. This non-capped power consumption is largely insensitive to imbalance.

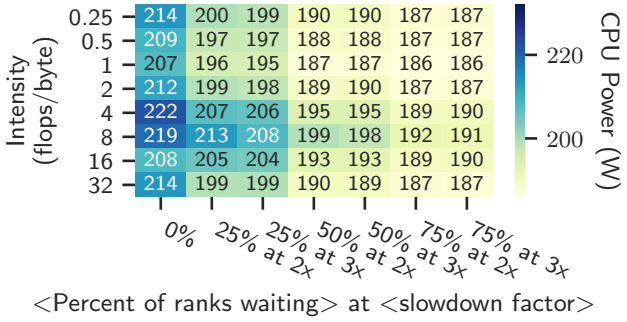

Fig. 5. Total CPU power per node for different workload configurations, when running the benchmark variant that uses 256-bit (ymm) vector registers under the GEOPM *power_balancer* agent. The most significant reductions in power from the *monitor* case are in the mid-intensity range. The apparent vertical bands indicate that the percent of waiting ranks have a strong effect on average needed power, which motivates using a performance-aware power management policy in those cases.

We obtain Metric-(b) by observing the actual power consumed by each workload when subjected to an average power budget equal to the total TDP (Thermal Design Power) of each node. The resulting measurements, obtained using the GEOPM *power_balancer* agent, are shown in Figure 5.

V. Experimental Setup

This section describes methods we follow to design our evaluation grid. This setup is divided into three sections. Section V-A explains the computing environment in which we run our experiments. Section V-B explains how we select combinations of workloads to evaluate in that environment. Section V-C outlines how we select power budgets to apply to those workload mixes.

We limit the scope of our investigation to focus on workload CPU power. Other work may explore a holistic data management system by also considering other major contributors to power and performance, such as network fabric, memory hierarchy, secondary storage, accelerator offloads, and cooling infrastructure. While we do not consider hardware variation in our comparisons, we control for it by selecting similarly-performing nodes for our experiments.

TABLE I
QUARTZ SYSTEM PROPERTIES

CPU	Intel Xeon E5-2695, dual-socket
Cores Per Node	36
Operating System	TOSS 3 (Red Hat Enterprise Linux 7.8)
Thermal Design Power	120 W per CPU socket
Minimum RAPL Limit	68 W per CPU socket
Base Frequency	2.1 GHz

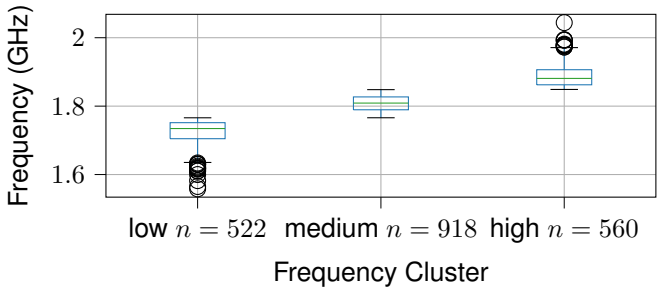

Fig. 6. Achieved frequencies of 2000 nodes in the Quartz cluster, under 70 watt CPU power limits. We use the *medium* frequency cluster for other stages of our experiments. Whiskers extend to the nearest samples beyond the inter-quartile range (IQR) by 1.5*IQR. Circles represent observations beyond the whiskers.

A. Test Environment

1) Quartz System: Our experiments are run on the LLNL Quartz system. The properties of Quartz are summarized in Table I. This system provides large-scale access to fine-grained power management knobs via the MSR-safe Linux Kernel module [13]. Our experiments utilize 34 cores per node for the benchmark, leaving the remaining two cores for monitoring processes and system services. Although these experiments are executed on a single Intel architecture, they can be ported to other architectures (Intel and non-Intel) by leveraging GEOPM's portable plugin infrastructure.

2) Hardware Variation: Many of our experiments are running under tight power constraints, so it is important to consider the impact of hardware variation when running under a power cap. To reduce the impact of hardware variation on our final results, we identified a subset of the cluster's nodes that perform similarly with our workload. To do this, we first monitored the achieved frequency of each node in the cluster while running our most power-hungry workload configurations under a low power limit. We used k-means clustering over the achieved frequencies to partition the nodes into three groups. We used the 918 *medium* frequency nodes in Figure 6 for our experiments so that our results reflect a central tendency of performance in a significant portion of our test system.

B. Workload Mixes

Our experiments include workload mixes that are composed of multiple configurations of the benchmark described in Section IV-A. The combinations of workloads in the mixes are summarized in Table II. We include some mixes that are expected to demonstrate best-case behavior for each of the baselines, as well as high power, low power, and random job mixes.

TABLE II
WORKLOADS IN EACH WORKLOAD MIX

		Workload Description	NeedUsedPower	HighImbalance	WastefulPower	LowPower	HighPower	RandomLarge
xmm (128-bit) vector registers	Balanced	No waiting ranks						
		0.25 FLOPs/byte	✗	✗	✗	✗	✗	✓
		16 FLOPs/byte	✓	✗	✗	✗	✗	✓
		32 FLOPs/byte	✓	✗	✗	✗	✗	✗
	2x Imbalance	25% waiting ranks						
		0.5 FLOPs/byte	✗	✗	✗	✗	✗	✓
		16 FLOPs/byte	✗	✗	✗	✓	✗	✗
		32 FLOPs/byte	✗	✗	✗	✓	✗	✗
		50% waiting ranks						
		0 FLOPs/byte	✗	✗	✗	✗	✗	✓
		0.25 FLOPs/byte	✗	✗	✓	✗	✗	✗
		0.5 FLOPs/byte	✗	✗	✓	✗	✗	✗
		1 FLOP/byte	✗	✗	✓	✗	✗	✗
		8 FLOPs/byte	✗	✗	✓	✗	✗	✗
		16 FLOPs/byte	✗	✗	✓	✗	✗	✗
		32 FLOPs/byte	✗	✗	✓	✓	✗	✗
		75% waiting ranks						
		32 FLOPs/byte	✗	✗	✗	✓	✗	✗
	3x Imbalance	25% waiting ranks						
		16 FLOPs/byte	✗	✗	✗	✓	✗	✗
		32 FLOPs/byte	✗	✗	✗	✓	✗	✓
		50% waiting ranks						
		16 FLOPs/byte	✗	✗	✓	✗	✗	✗
		32 FLOPs/byte	✗	✗	✓	✓	✗	✗
ymm (256-bit) vector registers	No Imbalance	No waiting ranks						
		0 FLOPs/byte	✗	✗	✗	✗	✓	✗
		0.25 FLOPs/byte	✗	✗	✗	✗	✗	✓
		4 FLOPs/byte	✗	✗	✗	✗	✓	✗
		8 FLOPs/byte	✗	✗	✗	✗	✓	✗
		32 FLOPs/byte	✓	✗	✓	✗	✗	✗
	2x Imbalance	25% waiting ranks						
		4 FLOPs/byte	✗	✗	✗	✗	✓	✗
		8 FLOPs/byte	✗	✗	✗	✗	✓	✗
		50% waiting ranks						
		8 FLOPs/byte	✗	✗	✗	✗	✓	✗
		75% waiting ranks						
		8 FLOPs/byte	✗	✗	✗	✗	✓	✗
	3x Imbalance	25% waiting ranks						
		0 FLOPs/byte	✗	✗	✗	✗	✗	✓
		0.25 FLOPs/byte	✗	✗	✗	✗	✗	✓
		2 FLOPs/byte	✗	✗	✗	✗	✗	✓
		8 FLOPs/byte	✗	✗	✗	✗	✓	✗
		32 FLOPs/byte	✗	✓	✗	✗	✗	✗
		50% waiting ranks						
		8 FLOPs/byte	✗	✗	✗	✗	✓	✗
		75% waiting ranks						
		16 FLOPs/byte	✗	✗	✗	✗	✗	✓

`NeedUsedPower` is expected to be the best-case mix for the `MinimizeWaste` policy. This case is characterized by some jobs with low average power and one job with high compute intensity, and where all used power is needed for performance, as determined by our *power_balancer* characterization runs. In this case, the `MinimizeWaste` policy is expected to decrease the power allocated to the many low-power jobs, and use that spare power to help the high compute

intensity job finish more quickly. Since the observed power is similar to the needed power for performance, policies that depend on performance-awareness are not expected to have an advantage.

`HighImbalance` is expected to be the best-case mix for the `JobAdaptive` policy. This case is characterized by a single, imbalanced job across all nodes. The `JobAdaptive` policy is expected to decrease the power to nodes within the single job that do not impact elapsed execution time, while increasing the power to nodes that do impact elapsed execution time. Since the workload is highly imbalanced within its bulk-synchronous loops, policies without performance awareness are not expected to have an advantage.

`WastefulPower` is expected to be the best-case mix for the `MixedAdaptive` policy. This is similar to the `NeedUsedPower` mix in the sense that it exhibits a range of average power levels. But it is different from that mix because the average power consumed by an unconstrained workload run significantly differs from the power consumed when the workload is balanced for performance under a generous power cap. The `MixedAdaptive` policy is expected to redistribute power within each job to maximize performance. As a result, it may reduce an imbalanced job's total allocated power to less than the total allocated power of `MinimizeWaste`, and it can share power across jobs unlike `JobAdaptive`.

`LowPower` contains the nine lowest power workload configurations, with 100 nodes per job.

`HighPower` contains the nine highest power workload configurations, with 100 nodes per job.

`RandomLarge` contains nine jobs selected from a random shuffle, with 100 nodes per job.

In order to identify the appropriate workload configurations for the above mixes, we separately characterize the power and performance characteristics of the workloads and of the hardware that runs our tests.

C. Selection of Power Budgets

We evaluate the power management policies at three different system-wide power budgets — *min*, *ideal*, and *max* — representing degrees of over-provisioning. The range of power caps we test covers the power capping region within which policies produce different power allocations relative to the needs of the workload mix. Power caps less than *min* result in all policies producing the same configuration as `StaticCaps`. Power caps greater than *max* result in all policies allocating at least as much power as `Precharacterized`. The quantitative values of the three power budgets are described in Table III.

The *min* power budget represents an aggressively over-provisioned system, where there is little capacity to share power across workloads. Its budget is selected by determining which workload in the mix has the least power consumed by a single node under the *performance-aware* characterization described in Section IV-B. The system is allocated enough power to provide that amount to each node.

TABLE III
POWER BUDGETS FOR EACH WORKLOAD MIX

Workload Mix	min	ideal	max*
NeedUsedPower	167 kW	171 kW	209 kW
HighImbalance	141 kW	163 kW	209 kW
WastefulPower	136 kW	144 kW	209 kW
LowPower	138 kW	152 kW	209 kW
HighPower	140 kW	177 kW	209 kW
RandomLarge	139 kW	164 kW	209 kW

*TDP of all CPUs is 216 kW

The *ideal* power budget represents over-provisioning that is ideal for a given workload mix. This budget is used in our experiments to evaluate cases where there is significant opportunity to improve performance by sharing power across workloads. Its limit is selected by summing the power used by each node for all workloads in the mix, as determined by the *performance-aware* characterization described in Section IV-B. The system is allocated that summed total of power.

The *max* power budget represents a conservatively over-provisioned system, where there is little need to share power across workloads. Its budget is selected by determining which workload in the mix has the most power consumed by a single node under the *uncapped* characterization described in Section IV-B. The system is allocated enough power to provide that much to each node.

VI. EXPERIMENTAL RESULTS

In this section, we evaluate the impact of various factors on elapsed time, energy, and power usage. Specifically, we discuss:

1) how system power limits impact performance under different policies
2) trade-offs of different levels of visibility into a system's resources and a workload's performance
3) how different workload mixes react to power management policies

A. Impact of System Power Cap

Support for hardware over-provisioning is one of the use cases for a system-wide power budget. We evaluate the impact of the system-wide power budgets described in Section V-C, which are meant to represent different levels of over-provisioning.

As shown in Figure 7, lower system-wide power limits, which correspond to more aggressively over-provisioned systems, result in closer to full utilization of the power capacity. The `Precharacterized` policy is unable to stay within the system-wide budget for all except the high power cap case, so it is omitted from further plots. Note that system-unaware policies may under-utilize the available power in cases with balanced supply and demand or with a surplus of power.

In cases of balanced power demand, annotated with marker-**(b)**, we can see that `JobAdaptive` is unable to utilize all of the budgeted power. This is because some budgeted power is set aside for workloads that cannot use all of their allocated

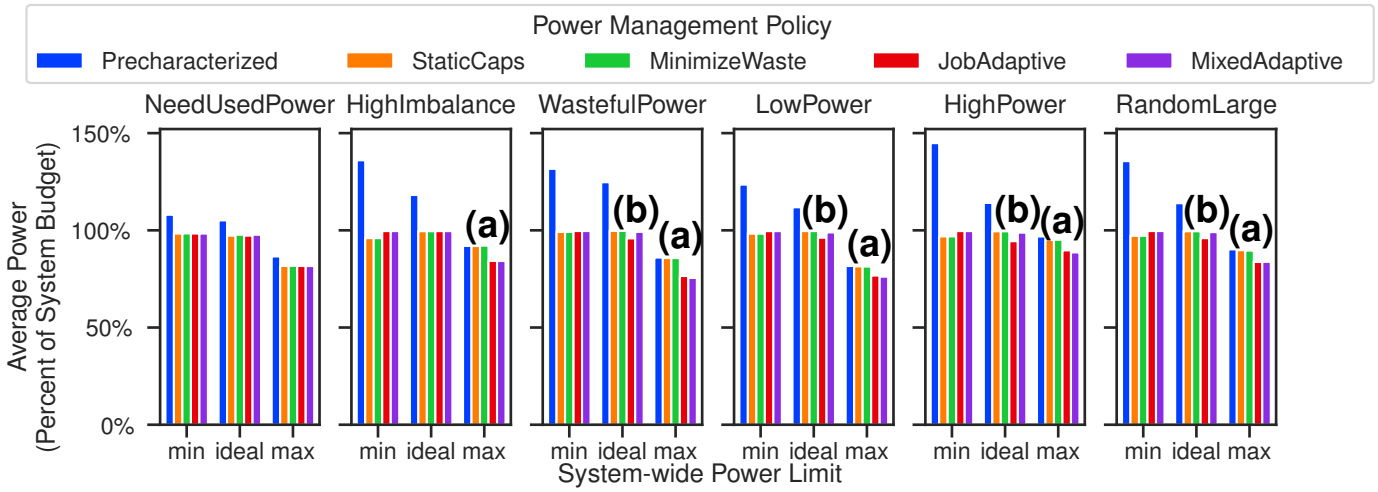

Fig. 7. Mean power used by each policy, across workload mixes (mixes described in Section V-B). Bars above 100% indicate that a policy exceeds the system-wide power budget. When bars are below 100%, that indicates either opportunity for more aggressive over-provisioning, or a missed opportunity to distribute power to places where it can be utilized. Annotations indicate cases where policies had large differences in power usage. Marker-**(a)** highlights max-power-limit cases where performance awareness enables less power usage under relaxed limits. Marker-**(b)** highlights ideal-power-limit cases where system power awareness enables more power utilization under a moderate limit.

power, while other workloads remain power-bound. The other policies are able to avoid this by using power awareness at the system level to share the budget across jobs.

In cases with high power budgets, annotated with marker-**(a)**, the amount of power being provisioned is higher than needed, regardless of performance-awareness in the policy. The difference between policies in these low-power-utilization cases is observable when we run the workloads and evaluate their energy impacts. In Section VI-B, we note that when job awareness enables reduction of a power budget, it results in improved energy efficiency. This efficiency improvement grows stronger at higher power limits where there is more opportunity to decrease power without impacting performance.

The impact of power caps is also visible as trends in mean time savings and energy savings in Figure 8. The time-saving opportunity decreases as system-wide power budget increases, with a maximum opportunity at 8% (marker-**(e)**) in the *min* power case for the `HighPower` workload mix. In that maximum opportunity scenario, the `MixedAdaptive` approach achieved about *7% time savings* over the `StaticCaps` baseline (marker-**(e)**). The energy-saving opportunity increases as power budget increases, with a maximum opportunity of *11% energy savings in the `WastefulPower` workload mix*, when subjected to the maximum power budget (marker-**(d)**).

Takeaway 1: Sites incorporating dynamic power management policies benefit from energy savings. These savings increase with the amount of surplus power budget.

B. Impact of Job Awareness

Our experiments evaluate a static power management policy as well as three policies that depend on workload characteristics, as outlined in Section III. Of the dynamic policies, only `JobAdaptive` and `MixedAdaptive` are aware of the performance of a running workload. Here, we note some takeaways about job awareness from traits that these policies share with each other, and not with the other polices.

The plots in Figure 8 show how energy, elapsed time, energy-delay product, and FLOPS per watt are impacted by each of the policies for different workload mixes. All metrics are reported as a percent improvement from the `StaticCaps` policy, which makes no workload-dependent changes to power allocation. The `Precharacterized` policy is omitted since it is unable to operate within the budgeted power in most cases.

One trend to note is that `JobAdaptive` and `MixedAdaptive` policies tend to perform similarly in the *min* and *max* power levels. Since the *min* power level has no opportunity for power sharing, both policies remain in their initial state, where power is distributed uniformly across hosts (refer to step 1 in Section III-A).

Those policies do have a chance to decrease their allocated power in the *max* power level case, which is why they are able to noticeably reduce total energy consumption in those runs. But, the cross-job power-sharing capability of the `MixedAdaptive` policy does not differentiate it from the `JobAdaptive` policy since there are no power-bound jobs that need to make use of that surplus.

Takeaway 2: HPC sites incorporating application awareness have increased opportunities for energy savings under a system power limit, compared to sites with application-agnostic solutions.

C. Impact of System-Wide Resource Awareness

The `MinimizeWaste` policy also depends on workload characteristics, but unlike `JobAdaptive` it is not aware of the workload's relationship between performance and power

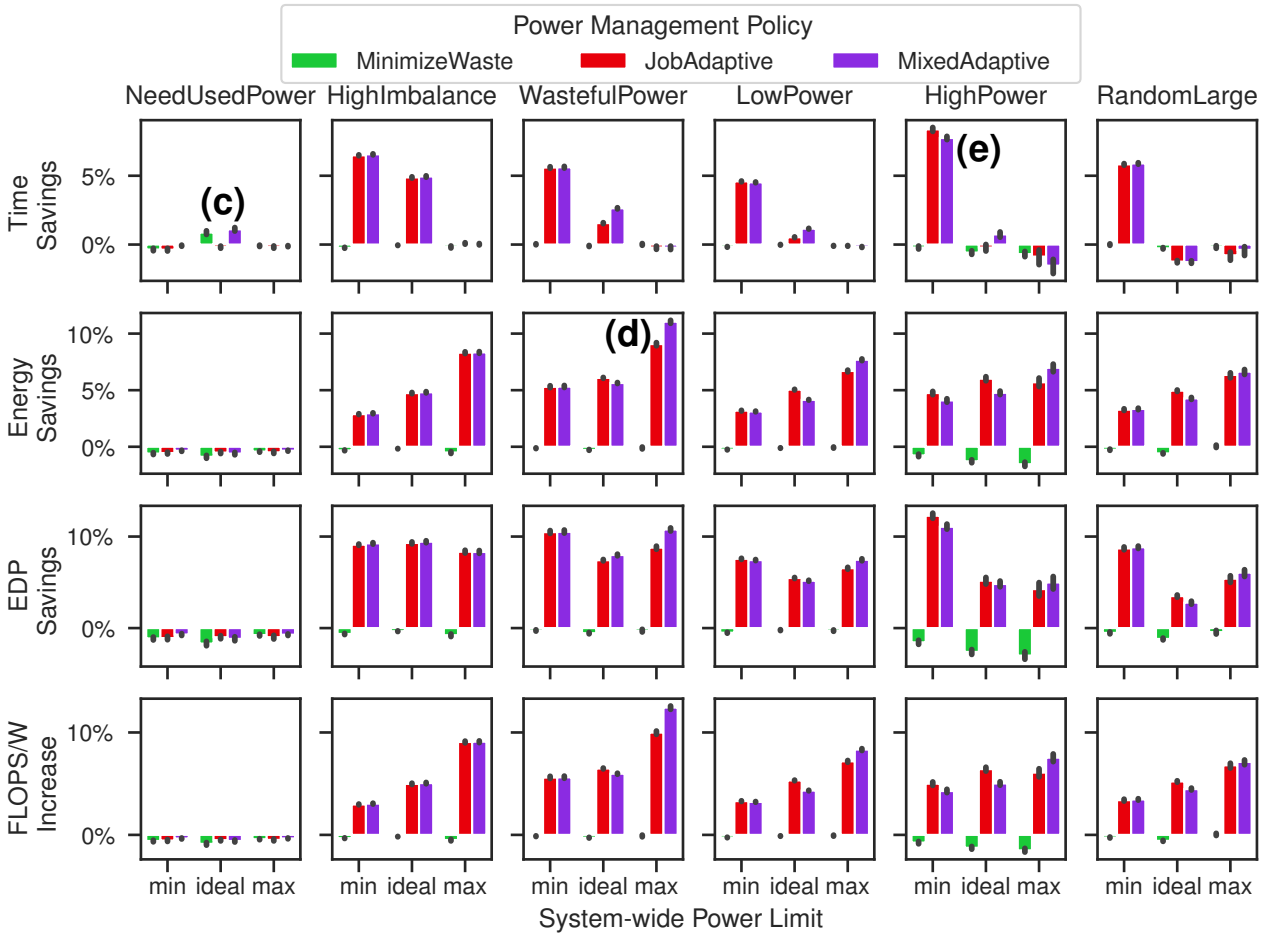

Fig. 8. The grid above depicts the impact of three system-wide power management policies (Section III) on different workload mixes (Section V-B) and power budgets (Section V-C). The height of each bar is quantified as a mean percentage decrease from the `StaticCaps` baseline measurements. Each row reports a metric of efficiency or performance. Each column reports a workload mix. Marker-**(c)** highlights an ideal-power-limit case where `MinimizeWaste` saves more time than `JobAdaptive`. Marker-**(d)** highlights a max-power-limit case where `MixedAdaptive` saves more energy than `JobAdaptive`. Marker-**(e)** highlights the case with the most time savings opportunity. Error bars show the 95% confidence interval, calculated over measurements from 100 iterations per benchmark configuration.

consumption. Instead, it relies only on observing and limiting power consumption from a resource manager's level of visibility in a cluster. `MixedAdaptive` also has this level of cluster power visibility, in addition to job awareness.

`MinimizeWaste` and `MixedAdaptive` both excel when workloads are all balanced with high compute intensity, and some workloads are more power-hungry than others. Unused power can be steered toward power-hungry workloads in that case, resulting in time savings for the `NeedUsedPower` workload mix in Figure 8.

The `MixedAdaptive` policy often performs similarly to the `JobAdaptive` policy. The cases where it stands out indicate scenarios where added resource awareness offers new opportunities. Cases that favor resource awareness manifest as additional energy savings when time savings are near equal. They are also visible in the form of time savings, as in the `WastefulPower` workload mix.

The energy savings of `MixedAdaptive` are further improved over those of `JobAdaptive` in some max power limit cases, most notably in the `WastefulPower` workload mix annotated with marker-**(d)**. These savings are a result of the final stage of the power allocation strategy used for these policies, described in Section III. After all needed power has been distributed to the workloads, the `JobAdaptive` policy continues to distribute the remainder power within each workload to the nodes that need the most power. In contrast, the `MixedAdaptive` policy uses its system-wide resource awareness to distribute the remaining power *across* all workloads to the nodes that need the most power. The `JobAdaptive` policy is not able to take the most efficient actions with the remainder of the power budget because it does not have system-wide power awareness.

Takeaway 3: Resource awareness alone has small benefits, but when combined with application awareness, can have greater benefits than either application awareness or resource awareness alone.

D. Impact of Workload Mixes

The running workloads significantly impact savings opportunities. While most of the mixes show significant time savings and energy savings for both `MixedAdaptive` and `JobAdaptive` policies, the `NeedUsedPower` mix shows no opportunity for energy savings, and small time savings for the `MinimizeWaste` and `MixedAdaptive` policies.

Both differences occur because the mix of jobs is composed of workloads that actually need all of their consumed power to avoid loss of performance. There is no opportunity to save energy because any reduction in power consumption will also cause the applications to take more time to finish running. Workloads under the `MinimizeWaste` policy can run faster as indicated by marker-**(c)** because the high power jobs are able to use the slack power budget from the low power jobs. The `JobAdaptive` policy can not see a benefit to elapsed time from re-balancing because the workloads are already balanced.

Takeaway 4: Application characteristics dictate the amount of surplus power available within for that application. Based on the mix of applications within a job schedule, this surplus power can be steered towards improving the efficiency of the entire system.

VII. Related Work

One of the key factors that differentiates this literature from past work is the incorporation of site-level, application-level, and node-level power management. To the best of our knowledge, there have been no past efforts within the HPC community that study the impact of all three layers in a unified manner. This aligns with this paper's position of advocating for interoperation between different levels of power management in a system stack. In this section, we group past works into categories that correspond to different layers of the stack.

A. Hardware-Level

Most hardware vendors provide some degree of firmware support for power management features. Examples of software interfaces to such features include DVFS (Dynamic Voltage Frequency Scaling) and power capping through vendor-specific interfaces such as Intel's RAPL (Running Average Power Limit) [14], IBM's PSR (Power Shifting Ratio) [15], NVML (NVIDIA Management Library) [16], and AMD's APM (Application Power Management) [17]. While power controls are exposed to software, the challenge is to choose the a configuration that ensures efficient or performant application execution. In this work, we integrate our power management solution with the GEOPM power-management framework that utilizes RAPL for monitoring and control.

B. Application-Level

Application-level power management techniques benefit from awareness of application characteristics and design patterns. Typically these approaches leverage monitor and control knobs exposed by the underlying hardware in addition to software-level controls like the number of threads, logical core count, environment variables, etc. Multiple efforts fall into these categories. These efforts include PUPiL by Zhang and Hoffmann, for power-constrained resource allocation [18], PShifter by Gholkar et al. which dynamically adjusts the ratio of compute to network time for improved energy efficiency [8], EAR by Corbalan et al. which detects application loops and scales frequency for reduced energy consumption [19]. Additionally, efforts like Adapt&Cap by Hankendi et al. [20] and the online algorithm [21] by De Sensi and Danelutto leverage software control knobs to boost efficiency.

All of these efforts support optimization algorithms that are application-aware, but remain oblivious of the site-specific requirements. In this work, we leverage GEOPM [4] to implement site-aware power management policies. These policies translate to specific algorithms which dynamically adapt to application requirements.

C. System-Level

State-of-the-art system-level power management techniques benefit from visibility into energy and power constraints of HPC facilities. Resource managers like HPE's Cray-ALPS [22] and Altair's PBS [23] offer static power management interfaces. Dynamic system-wide power management efforts like SchedMD's Slurm integrated power managment [3] and POW by Ellsworth et al. [2] demonstrate the utility of steering power from system components with low power demand to components with high power demand. The fundamental shortcoming of an application-agnostic solution is that the power management mechanism remains incapable of responding to application-level design characteristics.

The most similar prior work is Dynamo, by Wu et al. [24]. That paper proposes a framework which utilizes hardware power monitoring and controls at all layers of their data center's power delivery hierarchy. While Dynamo supports end-to-end power management, the framework relies on knowledge that specific nodes are limited to running particular workloads. It does not directly apply to typical shared HPC scenarios where compute nodes are subjected to previously-unseen workloads by multiple users. Our work captures variable power characteristics by ensuring that our search space includes a range of system power limits and different mixes of concurrently running applications.

In this paper, we compare different system-level policies with varying degrees of visibility to application properties and site-level power constraints. This strengthens our proposal for the community to work toward integrating system-level and application-level solutions.

VIII. Conclusion

While designing HPC sites under a power constraint, there arises a need to design a power management policy that works well in the common case, and minimizes the loss of quality of service in exceptional cases.

In this paper, we built upon existing efforts to use dynamic system-level power reallocation for greater power utilization of a cluster, and efforts to use dynamic workload-level power

reallocation to improve performance of a power-limited application. By combining these efforts, we show cases where each baseline policy can outperform the other. In this paper, we present an opportunity analysis highlighting the need for integrating application awareness with system-level resource awareness.

It is well-established that application-aware power management solutions can exhibit efficiency gains in a power-constrained environment. However, this approach alone fails to account for system-level power constraints that dictate HPC site policies. In this paper, we present empirical evidence that suggests that integrating application and system-level solutions together leads to more tangible savings, showing up to 7% reduction in system time and up to 11% savings in energy.

Since there is not currently an existing protocol or central mechanism for coordinating power management decisions across a data center's power delivery hierarchy, we emulated this execution time behavior by pre-characterizing our workloads and determining the steady-state power management properties ahead of time. By defining such protocol, this approach could be adapted to occur at execution time by coordinating system-level objectives of a resource manager with workload-level objectives of a job runtime.

As a follow-up to this literature, this work can further be extended to include additional levels of power and performance controls, such as cooling, networking, storage, and workload accelerators. Future work will also include extending this study to account for applications with multiple phases that have varying design characteristics. A large-scale empirical study for this effort will require extending modern-day job schedulers to dynamically react to changes in system-level policies. Through current and future efforts, we hope to establish a strong foundation for community-wide developments within the HPC PowerStack consortium.

ACKNOWLEDGMENTS

Development of the GEOPM software package has been partially funded through contract B609815 with Argonne National Laboratory. Part of this work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-CONF-815595).

REFERENCES

[1] J. Broughton, "HPC energy efficiency in the exascale era," https://eehpcwg.llnl.gov/assets/sc19_01_915_1000_hpc_energy_efficiency_in_the_exascale_era.pdf, Nov. 2019, keynote talk at the Energy Efficient HPC Workshop, SC 2019.

[2] D. A. Ellsworth, A. D. Malony, B. Rountree, and M. Schulz, "POW: System-wide dynamic reallocation of limited power in HPC," *HPDC 2015 - Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 145–148, 2015.

[3] "Slurm Power Management Guide," https://slurm.schedmd.com/power_mgmt.html, Mar. 2018, [Online; accessed 2020-01-28].

[4] J. Eastep, S. Sylvester, C. Cantalupo, B. Geltz, F. Ardanaz, A. Al-Rawi, K. Livingston, F. Keceli, M. Maiterth, and S. Jana, "Global extensible open power manager: A vehicle for hpc community collaboration on co-designed energy management solutions," in *High Performance Computing*, J. M. Kunkel, R. Yokota, P. Balaji, and D. Keyes, Eds. Cham: Springer International Publishing, 2017, pp. 394–412.

[5] M. Maiterth, G. Koenig, K. Pedretti, S. Jana, N. Bates, A. Borghesi, D. Montoya, A. Bartolini, and M. Puzovic, "Energy and power aware job scheduling and resource management: Global survey — initial analysis," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018, pp. 685–693.

[6] A. Marathe, Y. Zhang, G. Blanks, N. Kumbhare, G. Abdulla, and B. Rountree, "An empirical survey of performance and energy efficiency variation on intel processors," in *Proceedings of the 5th International Workshop on Energy Efficient Supercomputing*, ser. E2SC'17. New York, NY, USA: Association for Computing Machinery, 2017.

[7] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "Exploring hardware overprovisioning in power-constrained, high performance computing," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ser. ICS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 173–182.

[8] N. Gholkar, F. Mueller, B. Rountree, and A. Marathe, "Pshifter: Feedback-based dynamic power shifting within hpc jobs for performance," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 106–117.

[9] C. Cantalupo, J. Eastep, S. Jana, M. Kondo, M. Maiterth, A. Marathe, T. Patki, B. Rountree, R. Sakamoto, M. Schulz, and C. Trinitis, "A strawman for an hpc powerstack," 8 2018. [Online]. Available: https://hpcpowerstack.github.io/

[10] J. W. Choi, D. Bedard, R. Fowler, and R. Vuduc, "A Roofline Model of Energy," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. IEEE, may 2013, pp. 661–672.

[11] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, Apr. 2009.

[12] "Intel Advisor," https://software.intel.com/en-us/advisor, [Online; accessed 2020-03-04].

[13] "LLNL: msr-safe," https://github.com/LLNL/msr-safe, [Online; accessed 2020-04-02].

[14] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "Rapl: Memory power estimation and capping," in *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ser. ISLPED '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 189–194.

[15] P. S. Ratio, "Ibm power9 overview." [Online]. Available: https://variorum.readthedocs.io/en/latest/IBM.html

[16] NVIDIA, "Nvml api reference manual, vr450," June 2020. [Online]. Available: https://docs.nvidia.com/deploy/nvml-api/index.html

[17] AMD, "Linux tuning guide, amd opteron 6200 series processors," April 2012.

[18] H. Zhang and H. Hoffmann, "Maximizing performance under a power cap: A comparison of hardware, software, and hybrid techniques," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 545–559.

[19] J. Corbalan and L. Brochard, "Ear: Energy management framework for supercomputers," https://www.bsc.es/sites/default/files/public/bscw2/content/software-app/technical-documentation/ear.pdf, Barcelona Supercomputing Center (BSC), Tech. Rep., 2019.

[20] C. Hankendi, A. K. Coskun, and H. Hoffmann, "Adapt&cap: Coordinating system- and application-level adaptation for power-constrained systems," *IEEE Design & Test*, vol. 33, no. 1, pp. 68–76, 2016.

[21] D. De Sensi and M. Danelutto, "Application-aware power capping using nornir," in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, E. Deelman, J. Dongarra, and K. Karczewski, Eds. Cham: Springer International Publishing, 2020, pp. 191–202.

[22] HPE, "Xc(tm) series user application placement guide - s-2496." [Online]. Available: https://pubs.cray.com/bundle/XC_Series_User_Application_Placement_Guide_CLE60UP01_S-2496/page/Run_Applications_Using_the_aprun_Command.html

[23] Altair, "Pbspro administrator guide 2020." [Online]. Available: https://www.altair.com/pdfs/pbsworks/PBSAdminGuide2020.1.pdf

[24] Q. Wu, Q. Deng, L. Ganesh, C.-H. Hsu, Y. Jin, S. Kumar, B. Li, J. Meza, and Y. J. Song, "Dynamo: Facebook's data center-wide power management system," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. IEEE Press, 2016, p. 469–480.