An End-to-End HPC Framework for Dynamic Power Objectives

Daniel C. Wilson
danielcw@bu.edu
Boston University
Massachusetts, USA
Intel Corporation
Massachusetts, USA

Fatih Acun
acun@bu.edu
Boston University
Massachusetts, USA

Siddhartha Jana
Federico Ardanaz
Jonathan M. Eastep
siddhartha.jana@intel.com
federico.ardanaz@intel.com
jonathan.m.eastep@intel.com
Intel Corporation
Oregon, USA

Ioannis Ch. Paschalidis
Ayse K. Coskun
yannisp@bu.edu
acoskun@bu.edu
Boston University
Massachusetts, USA

ABSTRACT

High-Performance Computing (HPC) centers demand a lot of power, and continue to grow through the Exascale era. This work establishes the need for a multi-tiered, feedback-driven power management framework to follow dynamic power objectives while maximizing job performance, highlighting the need to respond to external factors (e.g., power constraints), and internal factors (e.g., performance variation). We present a practical implementation of this framework on a real-world cluster in addition to conducting simulations for larger data centers. We accurately track a moving power target for demand response while reacting to incomplete or inaccurate prior knowledge about job power and performance properties. We demonstrate that online performance feedback from a job runtime enables a cluster power management policy to recover most of the performance degradation introduced by job-type misclassification.

CCS CONCEPTS

• **Hardware → Enterprise level and data centers power issues**; **Impact on the environment**; • **Computing methodologies →** *Modeling methodologies.*

KEYWORDS

power management, demand response, high performance computing, data centers, quality of service

ACM Reference Format:
Daniel C. Wilson, Fatih Acun, Siddhartha Jana, Federico Ardanaz, Jonathan M. Eastep, Ioannis Ch. Paschalidis, and Ayse K. Coskun. 2023. An End-to-End HPC Framework for Dynamic Power Objectives. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3624062.3624262

1 INTRODUCTION

Data centers need to manage a lot of power, and they need to do so under time-varying constraints. Data center power consumption is estimated to account for 3% of the global energy supply [12] and reaches tens of megawatts for large facilities [23].

Data centers are exposed to time-varying constraints, such as different energy pricing based on time of day and peak consumption [5], or user requests that change over time. Data centers have previously been asked to be ready to participate in demand response events [13], in which the data center is asked to modify its power consumption to help balance supply and demand in an electrical grid. Participation in such programs helps increase grid flexibility, but the current rate of adoption of energy storage and demand response programs needs to double in order to offer sufficient grid flexibility for 2050 net-zero carbon emission goals [10]. It is important for data centers to be prepared to increase adoption of programs that require time-varying power management constraints, such as in demand response.

Prior work has demonstrated that data centers are suitable capacity providers in demand response programs, and have proposed policies to maximize cost savings while meeting a data center's quality of service (QoS) commitments to its users [3, 28, 29]. Those prior works assume that application power and performance properties are known in advance and do not change after profiling the applications.

We design a multi-tiered power management framework that enables a high degree of power sharing across jobs while being robust to scenarios where job performance diverges from precharacterized expectations. We measure missed performance opportunities in a

SC-W 2023, November 12–17, 2023, Denver, CO, USA

ACM ISBN 979-8-4007-0785-8/23/11…$15.00
https://doi.org/10.1145/3624062.3624262

power management policy that is purely driven by precharacterized job properties and demonstrate that our framework enables the power manager to recover the lost performance.

We implement and evaluate a cluster demand response policy on 16 real compute nodes. We explore the impacts of power management choices made when the power and performance properties of some jobs are unknown. Lastly, we simulate demand response scenarios with job performance variation[1] on a 1000-node data center to explore opportunities for further policy enhancements.

Key contributions of this work include:

- An end-to-end design framework to enforce cluster power policies while exploiting awareness of node-specific application properties.
- A practical implementation of the design framework, which delineates the power management at cluster-level from job-level policies. Management of the cluster-level tier is driven by awareness of the demand response policies within the job scheduler and resource manager. Management of the job-level tier is driven by awareness of application behavior and node-specific properties. This novel design enables integration of power management at both the tiers, without introducing excessive dependencies in either tier.
- An empirical evaluation that accounts for two types of unexpected job power-performance properties: The first is misclassification of pre-characterized job types, and the second is node-to-node performance variation.
- A discussion of using online performance feedback to respond to both of the performance modeling uncertainty cases above.

Through evaluations on a 16-node cluster, we show that our multi-level power management framework enables a demand response policy to stay within 8% of a time-varying cluster power target. We show that online performance feedback from a job-tier runtime enables a cluster-tier power manager to recover from performance degradation resulting from job-type misclassification. In simulations, we show QoS[2] degradation when the power manager is unaware of performance variation, and identify opportunities to integrate performance feedback.

The rest of this paper begins with a discussion of related work in multi-layer power management and in demand response for data centers in Section 2. Section 3 describes our multi-tiered power management framework, followed by a description of a practical implementation in Section 4. Section 5 describes the experimental methodology we follow to present results in Section 6. We discuss this work's outcomes and implementation challenges in Section 7 and discuss future work in Section 8. We conclude in Section 9.

2 RELATED WORK

In this section, we outline related works in performance-aware resource management, multi-tiered power management, and grid-aware scheduling and power management.

Various studies have investigated job power consumption in data centers to develop better power management and resource utilization policies while considering QoS constraints. A recent study demonstrates a power consumption analysis from the system, job, and user perspectives, and touches on using job queue metadata to predict job power properties [17]. Saillant et al. further investigate job power forecasting techniques, underlining the possible use case in cluster management policies [20]. Other studies introduce QoS-aware power management policies by considering job-level features [4, 6]. Those studies continuously monitor the performance of workloads to ensure whether they meet the QoS constraints. Our work aims to supplement forecasting efforts by responding to unknown or changing applications while they execute.

Some works consider methods to improve a job's performance under power constraints by steering power *within* a job based on performance feedback sent to a job-specific power manager [7, 9, 24]. Other methods improve power utilization by sharing power across jobs, albeit without reacting to measured job performance [8, 22]. Our proposed framework is intended to act as a platform to connect the two categories of approaches for feedback-driven end-to-end power management control.

In recent years, it has been well established within the HPC community that there exists a need for feedback-driven end-to-end power management control across the system stack. The HPC PowerStack Initiative [1, 2, 27] is working toward a standardized methodology for end-to-end data center power management from site down to hardware. They outline the architecture of system software within HPC systems that interact with each other to drive efficiency at different levels of granularity within the stack: at facility-level, system-level, node-level, job-level, and platform-level. Wilson et al. highlight opportunities to improve cluster efficiency through a collection of application characterization experiments that enable static power management policies based on application feedback [25]. Saba et al. propose a machine learning approach to predict job performance through feedback from performance counters while under CPU and GPU power caps, as part of a combined scheduling and power-capping solution [19]. Our work provides an empirical study that captures the benefits of incorporating power management across multiple tiers of the power management stack.

Wu et al. [26] propose a hierarchical approach for power management in data centers by reacting to power monitoring data at multiple hardware layers and pre-allocating servers for specific types of workloads. Another multi-tier power management solution focuses on increasing the power over-provisioning ratio of hyperscale data centers by exploiting the flexibility of non-production workloads [21]. In contrast, our work investigates HPC data center scenarios with a lower portion of non-production workload queues.

The ability to incorporate power management and task scheduling policies for data centers also provides the opportunity to mitigate the environmental and power-grid impact of data centers. The RMAP power manager accepts user tolerance for job slowdown as an input to a backfilling algorithm that sets static job power caps to improve throughput in overprovisioned cluster configurations [18]. Lots of studies underline the importance of task scheduling and power management for reducing data center carbon footprint and participation in demand response programs [11, 14, 15, 30].

[1] In this work, performance variation broadly refers to differences in job execution time from one run to another.

[2] QoS in this work corresponds to the time a user must wait for a submitted job to complete.

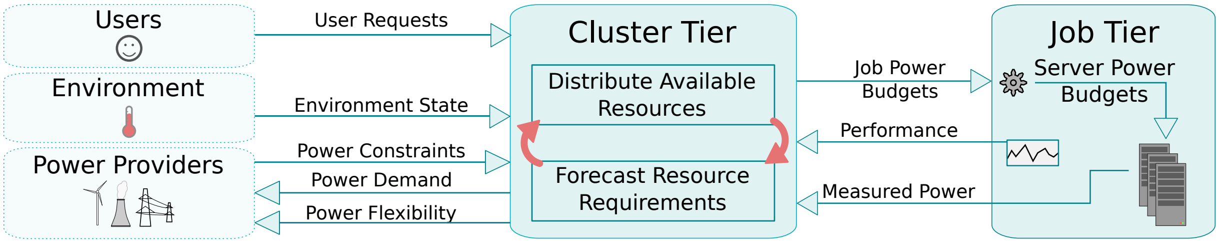

Figure 1: Tiers of power management entities in the ANOR framework. The cluster tier distributes the cluster's resources to members of the job tier. The cluster tier makes its decisions based on external state and aggregated power and performance data that comes from the job tier.

Our work introduces a holistic approach that brings together multi-tiered power management, power and performance-aware workload scheduling, and a demand response integration that allows data centers to cooperate with the grid to increase demand flexibility. By loosely coupling a cluster-tier power budgeting mechanism with a job-tier performance monitor, we enable end-to-end performance-aware power sharing across jobs in an HPC cluster.

3 A FRAMEWORK FOR MULTI-TIERED POWER MANAGEMENT: ANOR

Our goal is to design a framework that tracks time-varying cluster *power targets* under user QoS constraints where power-performance sensitivity may change over time. Potential applications for changing power targets include grid-aware power management scenarios where data center operators may react to time-varying carbon intensity, changing power tariffs, or demand response events. Time-varying QoS impacts may surface in many ways, such as seeing a new job type execute for the first time or running a previously seen job type under new conditions. In Section 4, we describe a specific implementation that works toward these goals to enable QoS-constrained demand response.

To that end, we propose *ANOR* (Attach Nested-Objective Runtimes), a multi-tiered design framework that manages a cluster with power objectives and QoS impacts that change over time. At a high level, this framework attaches two tiers of monitoring and control mechanisms, shown in Fig. 1. A cluster tier orchestrates entities in a job tier to optimize for application performance under time-varying constraints over total cluster power consumption. The tiers interact by sending control messages down to jobs from the cluster tier and by sending status messages up from the job tier.

This framework enables cluster power management policies to integrate performance awareness by delegating job-specific tasks to the job tier. We outline an implementation for demand response scenarios. Our cluster-tier policies for QoS-aware demand response are based on the **A**daptive policy with **Q**oS **A**ssurance (AQA) [29]. Our job-tier performance monitor utilizes the *GEOPM HPC runtime* [7]. Communication between the tiers is managed by TCP connections between the cluster manager and one compute-node process per job, which uses GEOPM's *endpoint interface* to monitor job progress and set power controls.

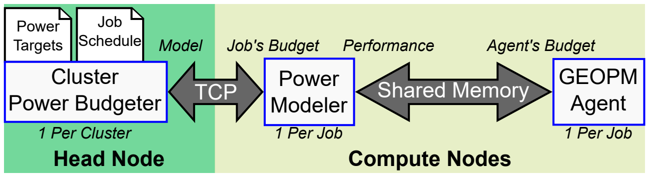

Figure 2: Our implementation of ANOR for demand response. A single cluster-tier process communicates over TCP with one job-tier power-modeling process per job, sending down power budgets and receiving power models. The power-modeling process sends power budgets to one GEOPM agent instance per job, over shared memory, and receives performance metrics back from the agent.

4 IMPLEMENTING ANOR FOR DEMAND RESPONSE

We demonstrate the ANOR framework by implementing it on the Global Extensible Open Power Manager (GEOPM) framework [7] to realize a performance-aware demand response policy. The demand response mechanism is based on the AQA [29] demand response bidder, job scheduler, and power budgeter.

We use the *GEOPM HPC runtime* to monitor and control node power consumption and to monitor job performance in our cluster. GEOPM provides *signals* to monitor applications (e.g., a count of times a region of code was entered) and hardware (e.g., power and energy), and provides *controls* for the hardware platform (e.g., CPU power caps). GEOPM offers a software framework to define *agents* that periodically read signals and write controls in response to those signals while a job executes. Agents on multi-node jobs interact across nodes through a hierarchical communication interface. The root level of that agent hierarchy has a software interface, called the *GEOPM endpoint interface*, that can be used to dynamically write new objectives and read summarized state updates from agents.

We implement a software layer that bridges the GEOPM endpoint to a job-tier power and performance model, illustrated in Fig. 2. This job-tier endpoint communicates through a TCP connection to a job-tier management process running on the head node in our cluster. The cluster-tier manager periodically reads cluster power targets from a file, receives messages from nodes running jobs, calculates how to distribute available power to jobs, and sends messages to inform each job-tier endpoint of the job's new power cap.

4.1 Cluster Power Budgeter

In this work, we consider two methods to allocate a cluster's available power to its compute nodes. We refer to the allocation mechanism as a power budgeter. An effective power budgeter ensures that the available power is allocated to components that use their power to improve system performance.

A single process balances the cluster's power budget from the head node in our cluster. For experimental repeatability (discussed in Section 5), this process reads power targets and a job submission schedule from files. Other implementations may schedule jobs through a separate scheduling mechanism, which is also compatible as long as new jobs initiate connections with the power budgeter.

Different power caps are selected by each of the two policies we investigate in this work when multiple jobs with different power-performance relationships run at the same time. The *power-balancing* approach aims for all jobs to operate on the same percent of power consumption across their range of achievable power. Each job has different performance impacts at that power ratio. In contrast, the *time-balancing* approach aims for every job to degrade performance by the same amount, but each job consumes a different amount of power.

4.2 Power Modeler

The cluster power budgeter uses power models to determine how to efficiently distribute a power budget. Each model relates a job's rate of progress to a CPU power cap. The modeler receives an *epoch* count (i.e., count of main loop iterations) from the GEOPM agent layer via the GEOPM endpoint interface. The modeler records the time since the last epoch update, and the average power cap applied over that time span. We fit $T = AP^2 + BP + C$ for T seconds per epoch and power cap P watts below TDP. We re-train the model when at least 10 new epochs have been recorded. Jobs that report no epochs or that have yet to build a model use a default model. Our contributions include an evaluation of performance impact from different default models (Section 6.1).

4.3 GEOPM Agent

The GEOPM agent enforces the budgeted power caps and feeds application performance to the modeler. The agent resides on the same node as the modeler, which communicates with the agent over shared memory through the GEOPM endpoint interface.

The agent reports the GEOPM *epoch count* to the endpoint interface. The epoch count is incremented each time all processes in the monitored program reach an instrumented `geopm_prof_epoch()` call in the program's main compute loop. We modified the GEOPM *power_governor* agent to write epoch count to the endpoint.

The endpoint supports multi-node jobs through GEOPM's internal agent communication layer. When the endpoint sends a new power cap to a job's GEOPM agent on one node, the agent forwards the power cap over a communication tree to the rest of the agent instances (one per node running the job).

4.4 Cluster-Tier Policies to Track Power Targets

The goal of our cluster policy is to achieve a target level of power consumption across the cluster while distributing performance loss across jobs. This policy enables performance-aware power capping without knowing the relationship between every job's power and performance in advance. We achieve that goal by delegating power-performance modeling to the job tier, as described in Section 4.

Cluster tier policies define how the job tier and external entities interact with each other. We explore policies that work with each other to meet QoS objectives while participating in demand response programs. A resource-forecasting policy aims to maximize the cluster's power flexibility under job performance constraints. A job scheduler and power capper maximize job performance under time-varying cluster power constraints.

4.4.1 Forecasting Power Demand. In our demand response scenario, the resource-forecasting policy determines how much *average power* the cluster should request and what range of power flexibility the cluster should offer as *reserve* for demand response. The bidding decision is made once per hour, influencing the range of power targets that will be received until the next bid. The data center must enforce time-varying power caps that span the average power plus or minus the reserve. New power targets arrive once every few seconds.

4.4.2 Scheduling Jobs. We base our scheduling and power management mechanisms on those introduced by the AQA policy [29]. AQA models job types as a collection of work queues. Each queue is assigned a weight of node allocations that is tuned over simulations of expected power-constraint and job-submission scenarios. Compute nodes are allocated so that so queues with greater weight are assigned more nodes.

AQA searches for queue weights and demand response bids (average power and reserve) that reduce electricity cost under constraints for QoS and power-tracking error. We set a power-tracking constraint allowing no more than 30% error for at least 90% of the time. Error is calculated as distance between the measured power and the target power, divided by the reserve. For example, if reserve is 100 kW and the absolute difference between actual power and target power is 10 kW, then the reported error is $10/100 = 10\%$. Power caps are applied uniformly across active nodes.

AQA assumes that all jobs are classified into precharacterized job types with known power-performance relationships. We use the existing job-weight training mechanism in AQA to support jobs with types that are not yet known at the time AQA is trained. For each unknown job type in the user submission queue during AQA training, we simulate a known minimum execution time (which may be provided at launch time, similar to setting a job's time limit). We simulate the job's achievable power-demand range and maximum slowdown (i.e., at the minimum power cap) to be randomly sampled from those of known job types.

4.4.3 Controlling Power Consumption. We evaluate two cluster-tier power-budgeting policies that utilize information from the job tier: a power-utilization balancer and a power-efficiency balancer.

The performance-unaware balancer follows the power capping rules used by the AQA [29]. In this approach, we select node power caps that are scaled equally between the minimum and maximum power achievable for all jobs. We select a single γ value that lets total power equal the power budget for:

$$p_{j,cap} = \gamma \left(p_{j,max} - p_{j,min}\right) + p_{j,min},$$

where each $p_{j,cap}$ is a job's selected power cap, $p_{j,min}$ is a job's minimum power consumption, and $p_{j,max}$ is a job's maximum power consumption.

The performance-aware balancer follows a similar approach, but balances the expected slowdown of each job instead of balancing the expected power consumption of each job. That is, we select the expected-slowdown limit s that lets the total power cap utilize the full power budget for:

$$p_{j,cap} = P_j\left(sT_j\left(p_{j,max}\right)\right),$$

where each P_j function models a job's power caps as a function of execution time, and each T_j function models that job's execution time as a function of power caps.

5 EXPERIMENTAL METHODOLOGY

Our experiments are designed with two goals in mind. First, we quantify the real-system impact that a multi-tiered power control mechanism has on a cluster. Next, we evaluate the implications of using such a mechanism at larger scale by simulating a 1000-node cluster. We perform both evaluations in demand response scenarios where the cluster is subjected to power targets that vary with time.

This section describes the benchmarks used in our evaluations, describes how we select QoS constraints, describes how we generate job schedules, describes the tools we use for performance and power measurement, and describes the system on which we execute our cluster experiments.

5.1 Benchmarks

We evaluate multiple combinations of concurrently executing benchmarks from the NAS Parallel Benchmark suite [16] across our simulated and real-cluster experiments. The power-performance relationship of each job type is shown in Fig. 3. The name of each job type follows a format of *benchmark-name.input-problem-class.process-count*.

This work investigates how cluster-level policies interact with job-level performance awareness, including scenarios with hour-long job submission schedules. We use benchmarks as placeholders to emulate different application phase characteristics. Multiple job schedules with mixes of these benchmarks emulate the variation of job behavior within job types concurrently executing in the cluster. Future works may investigate introducing additional complexity by also evaluating additional application phase variability within each job in a queue.

We insert a `geopm_prof_epoch()` function call once per iteration of the main outer loop in each benchmark, to indicate how long each iteration takes to complete. An *epoch count* is incremented after all processes across all nodes running the benchmark call this function. Each job's latest epoch count is reported whenever the cluster tier requests performance data from the job tier.

Our experiments use a combination of job power-performance curves that are precharacterized or learned at execution time. We precharacterize jobs by fitting them to the model described in Section 4.2. Most job types have training R^2 scores of at least 0.97. The exceptions are IS (0.92), MG (0.94), and SP (0.84).

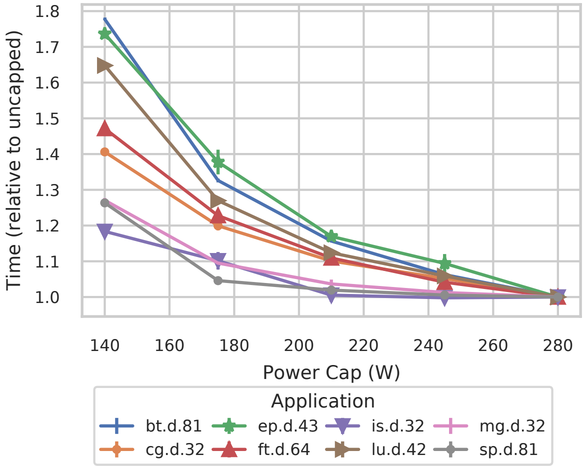

Figure 3: Execution time of each job type under varied power caps, relative to the execution time at a 280 W CPU power cap per node. Error bars show standard deviation over 10 runs.

5.2 QoS Constraint Selection

We use probabilistic QoS constraints as inputs to the scheduling and power-capping modules. We define a job's QoS degradation limit Q in terms of that job's sojourn time (i.e., elapsed time between job submission and job completion), T_{so}, and the amount of time that job takes to execute when it is not power limited, T_{min}:

$$Q = \frac{T_{so} - T_{min}}{T_{min}}.$$

QoS objectives should be configured for a data center based on the requirements of its users. We set the QoS objective for all job types in our experiments to be within $Q = 5$ with 90% probability. To justify this experimental design decision, we measured the queue wait time and execution time of jobs from a month of real-world job queue data [17]. The 90th percentile of job wait time divided by execution time is larger than 22, making our selected constraint more aggressive than the properties of that real-world queue trace.

5.3 Job Schedule Selection

We evaluate combinations of co-scheduled jobs across a range of power sensitivities, where power sensitivity indicates how much a job's performance changes under different power budgets. Some of our experiments (Section 6.2) do so through a randomly generated job schedule.

Job submissions are generated as Poisson processes with job arrival rates that achieve a target node utilization. We relate a target utilization η to job type j's arrival rate λ_j and non-power-capped time to completion T_j over N nodes by the following equation:

$$\sum_{j=1}^{J} \lambda_j T_j = \eta N.$$

5.4 Power and Performance Management

The cluster power manager periodically receives CPU power measurements from the CPU_ENERGY signal in the GEOPM HPC runtime. That signal aggregates energy across CPU package energy measurements from the PKG_ENERGY_STATUS MSR. The GEOPM agent enforces power targets with the CPU_POWER_LIMIT_CONTROL control in GEOPM, which maps to the PKG_POWER_LIMIT MSR. Our GEOPM instance uses MSRs through the msr-safe kernel module.

We measure application performance in two ways. For our hardware experiments, we report the time that the job spends running its benchmark, as reported by the *Application Totals* section of GEOPM reports that are generated for each job. For simulation experiments, we report the QoS metric described in Section 5.2.

5.5 Test Platform

The goals of our real-cluster experiments are two-fold. First, we aim to demonstrate that offline job performance model analyses sufficiently describe real opportunities in cluster power management. Then we evaluate the efficacy of ANOR as a way to mitigate job performance loss when offline analyses do not accurately represent an executing job. We run both evaluations on 16 nodes with dual-package Intel® Xeon® Gold 6152 CPUs and 100 GB RAM and 140 W Thermal Design Power (TDP) per socket.

At this scale, we can understand the range of individual application behavior as well as the impact of scheduling across multiple nodes with different jobs executing at the same time. This approach enables scaling to larger node counts by ensuring that at least one compute node per job can communicate with the cluster-tier power manager. Since this work requires modifications to job scheduler and resource manager, extending this study to larger node counts will require close collaboration with large-scale HPC centers.

5.6 Tabular Cluster Simulator

Some of our evaluations simulate a 1000-node cluster with random performance variation to evaluate the impact it has on QoS and power-tracking accuracy in demand response scenarios. Furthermore, we investigate whether alternative power-capping mechanisms can help mitigate any resulting impacts.

The simulator takes cluster and job-type properties, and produces a time series of cluster power consumption and a job queue with submission, start, and end time of each job. Input cluster properties include average idle power per node, total node count, average node utilization, and demand response parameters. Job type properties include the maximum acceptable QoS degradation as defined in Section 5.2, nodes per instance of the job type, maximum power per node while running the job, minimum power per node, and the elapsed execution time when the job runs with a cap at either of those power levels. Demand response parameters include average power $\bar{P}$, reserve power R offered by the simulated cluster, and a time-varying regulation signal $y(t)$. The regulation signal ranges from −1 to 1, indicating the cluster power target $P_{\text{target}} = \bar{P} + Ry(t)$.

The simulator is implemented as a collection of tables that store the current state of nodes and jobs in the cluster. The node table indicates whether a given node is idle, or which job it is executing, and tracks the current power consumption and current cap applied to each node. The job table keeps track of timestamps for queue entry, job start, and job end, as well as the type of job. The simulator also tracks the minimum and maximum power and time of each job type, to simulate a simple linear power-performance relationship.

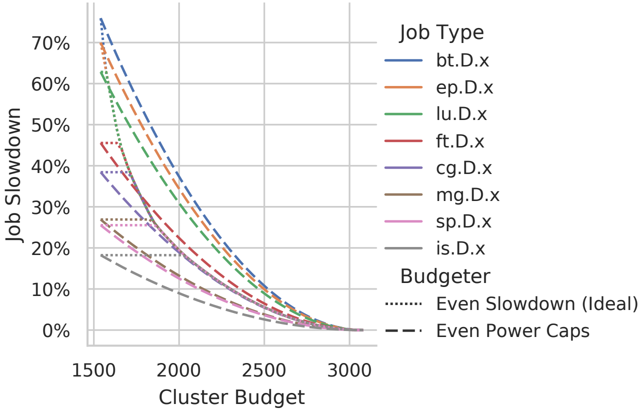

Figure 4: Estimated job slowdown when 8 job types each execute one instance under a range of shared power budgets.

Each simulated second, the simulator updates the state of the node table, then updates the view of the cluster seen by the job scheduler and power manager, then schedules jobs and caps power. The policy updates inputs to the node table that will be processed in the node-update stage of the next time step. Lastly, before starting the next iteration, we append the current state of all tables to a file.

Each non-idle node tracks its current job's progress. The rate of progress is calculated in each time step by linearly scaling between the job type's fastest and slowest precharacterized rate of progress, based on the server's current power cap. Some of our experiments introduce performance variation in the rate-of-progress by assigning a random multiplier to each simulated server for the duration of the simulation. We treat a simulated multi-node job as finished when all nodes executing the job have reached 100% progress.

6 RESULTS

In this section, we discuss the impacts of multi-tiered power management on the real cluster's power-tracking accuracy, and on the simulated cluster's demand response bidding and tracking accuracy.

6.1 Opportunities to partially precharacterize cluster job types

The goal of this set of experiments is to identify where job power-performance model accuracy is expected to impact job performance under a cluster power cap. We estimate the slowdown of each job under a given power cap, as modeled in Section 4.2. We evaluate the slowdown of jobs under ranges of cluster power caps following the performance-unaware (even power caps) and performance-aware (even slowdown) balancer policies described in Section 4.4.3.

6.1.1 Impact of integrating precharacterized performance models in a power budgeter. We estimate the expected slowdown of each job running in a cluster, assuming one of each job type in Section 5.1 is executing at the same time under a shared cluster power cap.

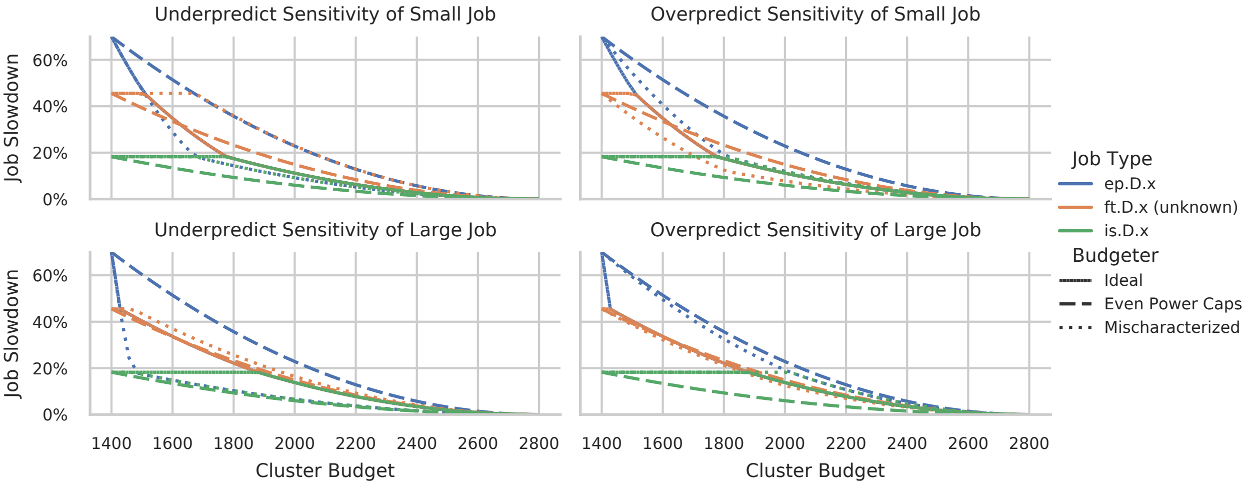

Figure 5: Performance impact when a medium-sensitivity job is misclassified as one with higher or lower sensitivity than its true behavior, while co-scheduled with both high-sensitivity and low-sensitivity jobs. Upper subplots model the unknown job as requiring more compute nodes than the known jobs, while lower subplots model the unknown job as requiring fewer compute nodes than the known jobs.

Fig. 4 shows the estimated slowdown of each job under a policy that distributes cluster power budgets for even slowdown across jobs and under a policy that distributes budgets for even power caps across nodes. As expected from Section 4.1, we see varied slowdown across jobs under the even-power budgeting policy, with a range of job slowdown that increases as the cluster budget decreases.

Under the *even-slowdown* policy, all jobs have equal expected slowdown at high cluster power budgets. As the cluster budget decreases, the jobs with lower power-performance sensitivity level off because they receive less power than high-power-sensitivity jobs in order to achieve equal slowdown. Those low-sensitivity jobs level off when they reach the system's minimum-allowed power cap (70 W per CPU package in our evaluation system).

The even-slowdown power-budgeting policy reduces the slowdown of the most severely impacted job across currently-scheduled jobs. There is no opportunity for reduced slowdown at minimum or maximum budgets since neither policy has flexibility to assign power caps beyond the range allowed by the power-capping interface. But there is opportunity in the mid-range power budgets.

6.1.2 Impact of misclassifying applications in a performance-aware power budgeter.

It may not be practical for a data center to require that all user jobs are characterized with power-performance models in advance of their execution. Some jobs may execute before they are characterized, or may be misclassified as a job type with different characteristics. This set of experiments evaluates the performance cost of misclassifying a job's power-performance sensitivity, and explores mechanisms that cope with misclassification.

Fig. 5 shows job slowdown in scenarios where low, medium, and high power sensitivity jobs are scheduled. Those jobs follow the power-performance curves of IS, FT, and EP, respectively. In these scenarios, the power-performance curve of FT is unknown to the power budgeter. The left subplots show performance of jobs if the budgeter assumes unknown jobs follow the sensitivity curve of the least-sensitive known job type (IS). The right subplots show the opposite policy where the budgeter assumes unknown jobs follow the curve of the most sensitive job type (EP). Upper subplots show cases where the unknown job is smaller (2 nodes) than the known jobs (4 nodes each), while lower subplots show cases where the unknown job is larger (8 nodes) than the known jobs (1 node each). The solid lines show the ideal slowdown when the budgeter knows the true sensitivity of all jobs, while the dashed lines show the slowdown if a performance-agnostic policy is employed, and dotted lines show the slowdown when the budgeter relies on a default sensitivity assumption for the unknown job type.

There are two key takeaways from comparing this set of scenarios. First, we note that an *underprediction policy* slows down the unknown job while an *overprediction policy* slows down the co-scheduled jobs that are highly power sensitive, compared to the ideal budgeter. This means that a power manager guided by precharacterized power-performance models must consider how to classify unknown job types based on whether it is more acceptable to degrade performance of the unknown job or other running jobs.

Second, the impact of misclassification depends on both the *size of the misclassified job* relative to the size of the co-scheduled jobs and the *direction of mischaracterization.* This should be accounted by a data center in order to minimize the risk of performance degradation when some jobs have not been precharacterized. Small unknown jobs are heavily slowed down when their power sensitivity is underpredicted. Large unknown jobs heavily slow down other executing jobs when the unknown job's sensitivity is overpredicted.

Regardless of which sensitivity assumption is used for unknown job types, there is a risk of misclassification, which will either limit the performance of the unknown job or the other jobs running in the cluster. In case of misclassification, it is important to select a policy for unknown job types that exposes whichever type of cluster performance risk is preferred in the data center, or to have a method to detect the misclassification and adjust the power budget.

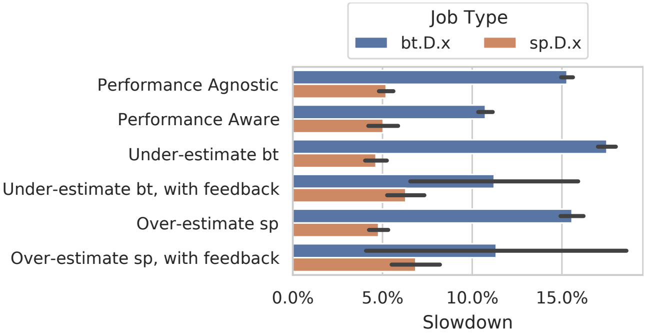

Figure 6: Job slowdown when BT (high power sensitivity) and SP (low power sensitivity) are co-scheduled under a shared power budget with 75% of TDP available in the budget. Slowdown is reported as a percentage of each job type's time to completion when there is no power cap. Error bars show standard deviation over 3 trials of SP.

6.2 Performance Under Shared Power Caps on Real Hardware

The goal of this set of experiments is to demonstrate how the offline analysis results translate to jobs executing under a power cap on a real cluster. In these experiments, we measure the slowdown of jobs under different mixes of co-scheduled job types. Specifically, we consider 3 scenarios where there are two jobs with low power sensitivity (both SP), two jobs with high power sensitivity (both BT), or one each of high and low (BT and SP) power sensitivity job types.

Under these 3 job-type scenarios, we measure each job's execution time under a static power budget that is shared across 4 nodes by a power-balancing policy and by a performance-balancing policy that relies on precharacterized power-performance curves to distribute the shared power cap. Here, we set the budget to 840 W, mid-way between the maximum and minimum power caps supported by our test platform. We consider cases where each job's type is known in advance, so the power balancer selects the correct performance model, as well as cases where the job's type is mispredicted to be one of much higher or much lower power sensitivity.

Fig. 6 shows the job slowdown under multiple scenarios of model accuracy given a job mix that includes one instance each of BT and SP executing in the cluster at the same time. As expected from the offline model analysis, the performance of power-sensitive BT degrades from the fully characterized case both when that job is misclassified as low sensitivity, or when its co-scheduled job is misclassified as high sensitivity. Furthermore, by allowing the system to re-train a job's model when performance feedback is available, the power budgeter reduces the slowdown of BT in both misclassification cases.

Although the fully characterized case does not completely close the slowdown gap between BT and SP, it does reduce the gap in comparison to the performance-agnostic policy. One way this might be addressed is by improving the accuracy of the characterization model. A more robust control system may also help by reacting to modeling error as part of including performance feedback.

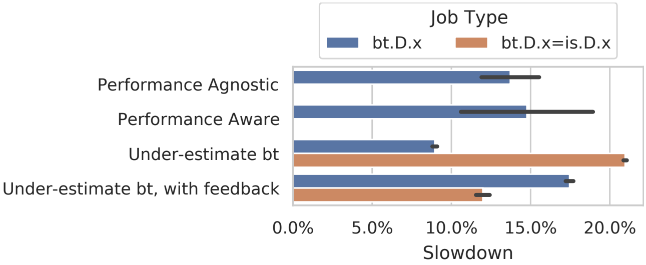

Figure 7: Job slowdown when two instances of BT (high power sensitivity) are co-scheduled under a shared power budget with 75% of TDP available in the budget, with one instance potentially being misclassified as IS. Slowdown is reported as a percentage of each job type's time to completion when there is no power cap. Error bars show standard deviation over 3 back-to-back trials.

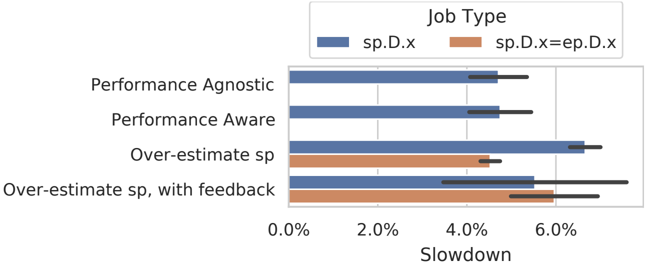

Figure 8: Job slowdown when two instances of SP (low power sensitivity) are co-scheduled under a shared power budget with 75% of TDP available in the budget, with one instance potentially being misclassified as EP. Slowdown is reported as a percentage of each job type's time to completion when there is no power cap. Error bars show standard deviation over 6 back-to-back trials.

Figs. 7 and 8 both show that performance-agnostic solutions perform similar to the precharacterized power budgeter when all scheduled jobs have similar power-performance properties. Since the scheduled job types have the same power-performance tradeoffs, both solutions make the same decisions. In Fig. 7, we see increased slowdown when a high-sensitivity job type is misclassified. By misclassifying one of the low-sensitivity jobs in Fig. 8, we see a small slowdown for its co-scheduled job. In both cases, we are able to recover some of the lost performance by communicating information about the unexpected slowdown from the job tier to the cluster tier's power budgeter.

6.3 Performance Under Time-Varying Power Caps

In this section, we evaluate job performance under a time-varying power cap through a 1-hour job schedule. The power target changes once every 4 seconds, staying within the range of 2.3 kW to 4.5 kW, as shown in Fig. 9. Our power objective is not just to stay less than the power target, but to closely follow the power target.

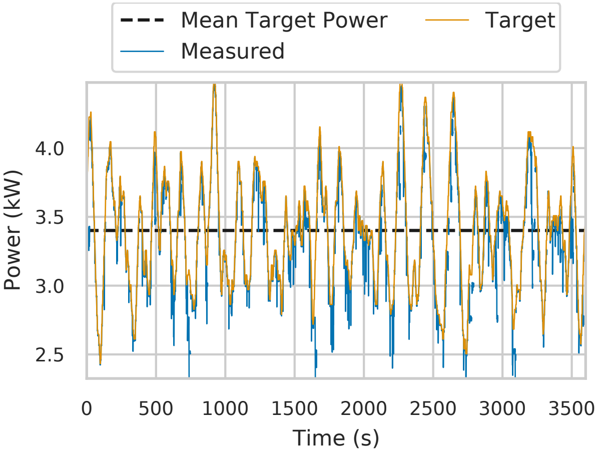

Figure 9: Time-varying cluster power targets and measurements using ANOR over an hour of job arrivals from 6 job types. The y axis spans the cluster's committed range of power flexibility around the requested mean target power.

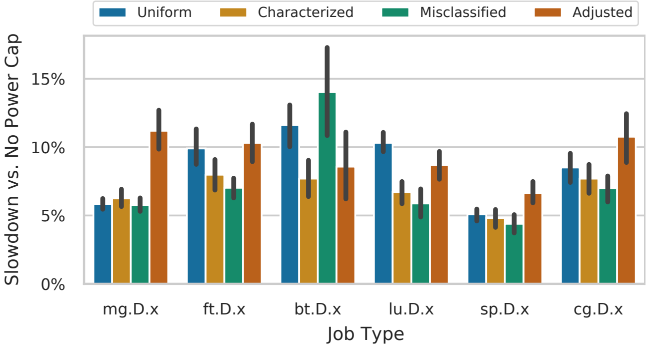

Figure 10: Mean execution-time slowdown of job types under a 1-hour schedule with time-varying cluster power caps. Slowdown is shown as a percentage of each job type's mean execution under no power cap. Error bars indicate the 95% confidence interval.

Fig. 10 shows the execution-time impact of different power capping techniques summarized across 6 job types arriving for 95% node utilization as described in Section 5.3, using the scheduler described in Section 4.4.2. These results show that the slowdown of jobs with greater power sensitivity (BT, LU, and FT) under a uniform power distribution policy is greater on average than the slowdown of other co-scheduled jobs. The performance-aware (characterized) balancer improves the worst-job slowdown by steering power toward power-sensitive jobs. As a result, the three most-sensitive job types achieve less slowdown on average, reducing the slowest job type from 11.6% slowdown to 8.0% slowdown.

The misclassified runs represent the case where BT (high power sensitivity) is misclassified as IS (low power sensitivity). This slows down BT on average, but performance feedback from the job tier enables the adjusted policy to recover much of the lost performance.

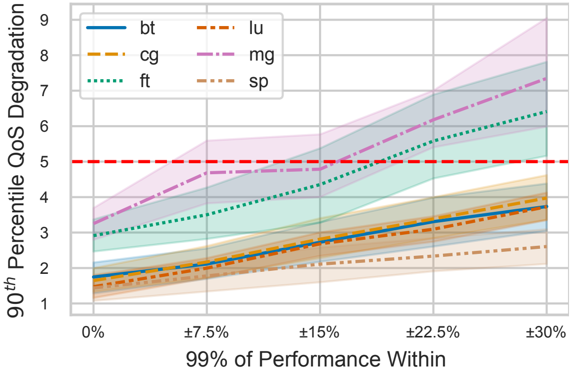

Figure 11: QoS degradation under different levels of performance variation. Each line presents the mean degradation over 10 trials. The shaded region indicates the 90% confidence interval. The horizontal dashed line indicates the QoS target of all applications.

Measured power is under 24% error at least 90% of the time in the worst case (misclassified, without job feedback), within our error constraint. All other cases are under 17% error.

6.4 Impact of Performance Variation on QoS Degradation

We evaluate our policy under different levels of performance variation to observe the impact of variation on the QoS degradation of jobs. To that end, we generate performance coefficients from a normal distribution with a mean of 1, and adjust the standard deviation to change the level of performance variation. The performance coefficients are randomly generated for each of 1000 compute nodes at the start of each of 10 simulations per variation level. Each simulation uses a different random seed that impacts performance coefficients and job arrival times of 6 job types at 75% utilization. All jobs are scaled to use 25x as many nodes as the job types used in the 16-node cluster experiments. Under each level of performance variation, our method's power tracking error is within our constraint for less than 30% error 90% of the time.

Fig. 11 shows the 90th percentile of QoS degradation across performance variation levels. The results indicate that, across all applications, a greater degree of performance variation causes more QoS degradation. However, since different applications have varying levels of sensitivity to performance variation, some of them more rapidly degrade beyond the QoS threshold of 5. While we are able to avoid capping power on jobs that application feedback indicates are at risk of QoS degradation, we found that this does not improve QoS significantly because the AQA policy already lightly caps power on jobs in this schedule, primarily reducing power by refraining from scheduling jobs to idle nodes. Future work may place more emphasis on node power capping, which can be adjusted after jobs are already scheduled. Such a decision reduces the range of power control that AQA can offer for demand response, but may be a reasonable trade-off under tight QoS constraints.

7 DISCUSSION

In this work, we evaluate the cluster-power-tracking accuracy and job QoS impacts of a multi-layered power capping method, and we investigate the opportunities and challenges that a multi-layered solution introduces to a demand response power management scenario. In this section, we discuss alternative scopes of power management and outline some practical challenges we encountered.

7.1 Scope of Power Management

We limit our investigation to CPU power monitoring and control to simplify the initial implementation. The same framework can be used with wider scope through modeling or additional monitoring. For example, the job tier can model or directly monitor more devices and components in the nodes. Alternatively, the cluster tier can apply a model of its total power demand as a function of the job tier's power and other state within the cluster (e.g., environmental conditions). A data center facility may use its own models and metering infrastructure directly for higher-level power monitoring.

7.2 Practical Implementation Challenges

We encountered practical challenges while implementing and evaluating a performance-aware power-capping system. Specifically, we encountered challenges with asynchronous management of samples and summarized metrics across power management tiers, and challenges transitioning from ideal and static cluster scenarios to end-to-end scheduled scenarios.

For asynchronous sample management, we initially needed to gather many samples from the job runtime to consistently map power caps to job performance metrics. To tackle this issue, we introduced timestamps so that different tiers of the power management system can be mapped to each other when the tiers are not executing their control loops at the same rate.

We initially observed unexpected performance *improvements* in all power management policies when we moved from ideal power-capping scenarios to real-world experiments. Ultimately, those improvements ended up being due to two job types (IS and EP) that have very short execution times (less than half a minute). The time spent setting up and tearing down those short jobs (both in the batch system and in the job itself) represents a major share of the total time those jobs hold compute node resources in the batch submission system. During that time, the compute node's power consumption is low, which enables all policies to reallocate extra slack power to all other active jobs for most of the time the short job is active. We omit those job types from schedules in our final evaluation since they hide the slowdown that would be expected in a system with mostly minutes-or-longer jobs.

8 FUTURE WORK

Multi-tier power management is rich with opportunities for future work to investigate scalability challenges, harnessing additional control levels within tiers, and additional application usage patterns.

Multi-tier power management may introduce scalability challenges when there are many concurrent jobs. Our example implementation limits the depth of control needed in the widely-scoped cluster tier by delegating application-aware and node-aware processing to the job tier. The cluster tier needs to know the coefficients of the job performance model to distribute power effectively, and it needs to issue new power caps to all jobs whenever the distribution shifts. Future work may investigate methods to reduce the required communication or to localize it within additional control hierarchy.

In addition to investigating scalability within a cluster, there are opportunities to scale beyond a single cluster. For example, a facility with multiple clusters may wish to coordinate power demand across those clusters. Our proposed framework may be extended by treating the facility as a power provider to each member of the cluster tier. Coordinated power management across clusters could be particularly useful for facilities that are bringing up next-generation clusters while previous-generation clusters are still operating under a shared power infrastructure that may not have the capacity to use both clusters at peak power demand concurrently.

Modern HPC systems have software controls for more than just CPU power (e.g., accelerator power caps, node power caps, frequency and voltage controls). Current works already investigate methods to combine CPU power controls with other power controls in a node. Additional work may consider harnessing more control levels within power management tiers. For example, the cluster tier may distribute node power caps across jobs based on coarse job characterization models, but the job tier may locally explore power and performance trade-offs across resources within jobs.

Our example multi-tiered power management implementation utilizes a single characterization per job, informed by manually-inserted instrumentation. However, some jobs may consist of multiple power-sensitivity profiles through the job's lifecycle, or may even contain multiple concurrent profiles executing at a time (e.g., for simulations running alongside in-situ data analyzers). Future work may consider how to handle job phase changes across the management hierarchy, and how to handle multiple distinct phases concurrently within a job. Such efforts will also benefit from reusable instrumentation (e.g., by annotating commonly-used software libraries) or automatic epoch detection (e.g., by identifying periodic usage of system resources or software interfaces).

9 CONCLUSION

In this work, we motivate the need for feedback-driven multi-tiered cluster power management stack. We present ANOR, which couples power management frameworks at cluster level and job level in order to achieve system-wide power and performance constraints. Our analysis reveals trends where job performance awareness is expected to improve system performance under a cluster power cap versus a performance-agnostic policy, but that performance degrades when pre-characterization data is missing or inaccurate. We demonstrate a practical implementation of a QoS-aware data center demand response policy through the ANOR framework, successfully tracking power in many cases, and identifying practical challenges for others. Furthermore, we show a performance variation analysis in 1000-node cluster simulations.

10 ACKNOWLEDGMENTS

Development of the GEOPM software package has been partially funded through contract B609815 with Argonne National Laboratory. Some of this work was partially funded by the Hariri Institute and the Institute for Global Sustainability at Boston University.

REFERENCES

[1] Eishi Arima, A. Isaías Comprés, and Martin Schulz. 2022. On the Convergence of Malleability and the HPC PowerStack: Exploiting Dynamism in Over-Provisioned and Power-Constrained HPC Systems. In *High Performance Computing. ISC High Performance 2022 International Workshops*, Hartwig Anzt, Amanda Bienz, Piotr Luszczek, and Marc Baboulin (Eds.). Springer International Publishing, Cham, 206–217.

[2] Christopher Cantalupo, Jonathan Eastep, Siddhartha Jana, Masaaki Kondo, Matthias Maiterth, Aniruddha Marathe, Tapasya Patki, Barry Rountree, Ryuichi Sakamoto, Martin Schulz, and Carsten Trinitis. 2018. A Strawman for an HPC PowerStack. (8 2018). https://doi.org/10.2172/1466153

[3] Hao Chen, Yijia Zhang, Michael C. Caramanis, and Ayse K. Coskun. 2019. EnergyQARE: QoS-Aware Data Center Participation in Smart Grid Regulation Service Reserve Provision. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 4, 1, Article 2 (jan 2019), 31 pages. https://doi.org/10.1145/3243172

[4] Shuang Chen, Angela Jin, Christina Delimitrou, and José F. Martínez. 2022. ReTail: Opting for Learning Simplicity to Enable QoS-Aware Power Management in the Cloud. In *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, Seoul, Korea, 155–168. https://doi.org/10.1109/HPCA53966.2022.00020

[5] Anders Clausen, Gregory Koenig, Sonja Klingert, Girish Ghatikar, Peter M. Schwartz, and Natalie Bates. 2019. An Analysis of Contracts and Relationships between Supercomputing Centers and Electricity Service Providers. In *Workshop Proceedings of the 48th International Conference on Parallel Processing* (Kyoto, Japan) *(ICPP Workshops '19)*. Association for Computing Machinery, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3339186.3339209

[6] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-efficient and QoS-aware cluster management. *ACM SIGPLAN Notices* 49, 4 (2014), 127–144.

[7] Jonathan Eastep, Steve Sylvester, Christopher Cantalupo, Brad Geltz, Federico Ardanaz, Asma Al-Rawi, Kelly Livingston, Fuat Keceli, Matthias Maiterth, and Siddhartha Jana. 2017. Global Extensible Open Power Manager: A Vehicle for HPC Community Collaboration on Co-Designed Energy Management Solutions. In *High Performance Computing*, Julian M. Kunkel, Rio Yokota, Pavan Balaji, and David Keyes (Eds.). Springer International Publishing, Cham, 394–412.

[8] Daniel A. Ellsworth, Allen D. Malony, Barry Rountree, and Martin Schulz. 2015. 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* (2015), 145–148. https://doi.org/10.1145/2749246.2749277

[9] Neha Gholkar, Frank Mueller, Barry Rountree, and Aniruddha Marathe. 2018. 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* (Tempe, Arizona) *(HPDC '18)*. Association for Computing Machinery, New York, NY, USA, 106–117. https://doi.org/10.1145/3208040.3208047

[10] IEA 2022. *Demand Response*. IEA. Retrieved March 14, 2023 from https://www.iea.org/reports/demand-response

[11] Ali Jahanshahi, Nanpeng Yu, and Daniel Wong. 2022. PowerMorph: QoS-aware server power reshaping for data center regulation service. *ACM Transactions on Architecture and Code Optimization (TACO)* 19, 3 (2022), 1–27.

[12] Bran Knowles. 2021. ACM TechBrief: Computing and Climate Change. *ACM Technology Policy Council* (Nov. 2021).

[13] Jacklin Kwan. 2022. Climate change threatens supercomputers. *Science (New York, NY)* 378, 6616 (2022), 124–124. https://www.science.org/doi/10.1126/science.adf2882

[14] Zhenhua Liu, Yuan Chen, Cullen Bash, Adam Wierman, Daniel Gmach, Zhikui Wang, Manish Marwah, and Chris Hyser. 2012. Renewable and Cooling Aware Workload Management for Sustainable Data Centers, In Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems. *SIGMETRICS Perform. Eval. Rev.* 40, 1, 175–186. https://doi.org/10.1145/2318857.2254779

[15] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. 2013. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. In *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*. Elsevier, New York, NY, USA, 341–342.

[16] nasa.gov. 2022. NAS Parallel Benchmarks. https://www.nas.nasa.gov/software/npb.html.

[17] Tirthak Patel, Adam Wagenhäuser, Christopher Eibel, Timo Hönig, Thomas Zeiser, and Devesh Tiwari. 2020. What does power consumption behavior of hpc jobs reveal?: Demystifying, quantifying, and predicting power consumption characteristics. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, IEEE, New Orleans, LA, USA, 799–809.

[18] Tapasya Patki, David K. Lowenthal, Anjana Sasidharan, Matthias Maiterth, Barry L. Rountree, Martin Schulz, and Bronis R. de Supinski. 2015. Practical Resource Management in Power-Constrained, High Performance Computing. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing* (Portland, Oregon, USA) *(HPDC '15)*. Association for Computing Machinery, New York, NY, USA, 121–132. https://doi.org/10.1145/2749246.2749262

[19] Issa Saba, Eishi Arima, Dai Liu, and Martin Schulz. 2022. Orchestrated Co-scheduling, Resource Partitioning, and Power Capping on CPU-GPU Heterogeneous Systems via Machine Learning. In *Architecture of Computing Systems*, Martin Schulz, Carsten Trinitis, Nikela Papadopoulou, and Thilo Pionteck (Eds.). Springer International Publishing, Cham, 51–67.

[20] Théo Saillant, Jean-Christophe Weill, and Mathilde Mougeot. 2020. Predicting job power consumption based on rjms submission data in hpc systems. In *High Performance Computing: 35th International Conference, ISC High Performance 2020, Frankfurt/Main, Germany, June 22–25, 2020, Proceedings 35*. Springer, Springer, Cham, Frankfurt/Main, Germany, 63–82.

[21] Varun Sakalkar, Vasileios Kontorinis, David Landhuis, Shaohong Li, Darren De Ronde, Thomas Blooming, Anand Ramesh, James Kennedy, Christopher Malone, Jimmy Clidaras, et al. 2020. Data center power oversubscription with a medium voltage power plane and priority-aware capping. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. Association for Computing Machinery, Lausanne, Switzerland, 497–511.

[22] Slurm Power Management Guide 2018. Slurm Power Management Guide. https://slurm.schedmd.com/power_mgmt.html. [Online; accessed 2022-06-13].

[23] TOP500 List - June 2023 2023. TOP500 List - June 2023. https://www.top500.org/lists/top500/list/2023/06/.

[24] Daniel C. Wilson, Asma H. Al-rawi, Lowren H. Lawson, Siddhartha Jana, Federico Ardanaz, Jonathan M. Eastep, and Ayse K. Coskun. 2022. Guiding Hardware-Driven Turbo with Application Performance Awareness. In *2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC)*. 1–8. https://doi.org/10.1109/IGSC55832.2022.9969356

[25] 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, and Ayse K. Coskun. 2021. Introducing Application Awareness Into a Unified Power Management Stack. In *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 320–329. https://doi.org/10.1109/IPDPS49936.2021.00040

[26] Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang-Hong Hsu, Yun Jin, Sanjeev Kumar, Bin Li, Justin Meza, and Yee Jiun Song. 2016. Dynamo: Facebook's data center-wide power management system. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 469–480.

[27] Xingfu Wu, Aniruddha Marathe, Siddhartha Jana, Ondrej Vysocky, Jophin John, Andrea Bartolini, Lubomir Riha, Michael Gerndt, Valerie Taylor, and Sridutt Bhalachandra. 2020. Toward an End-to-End Auto-tuning Framework in HPC PowerStack. In *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. 473–483. https://doi.org/10.1109/CLUSTER49012.2020.00068

[28] Yijia Zhang, Daniel C. Wilson, Ioannis Ch. Paschalidis, and Ayse K. Coskun. 2021. A Data Center Demand Response Policy for Real-World Workload Scenarios in HPC. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, Grenoble, France, 282–287. https://doi.org/10.23919/DATE51398.2021.9474075

[29] Yijia Zhang, Daniel Curtis Wilson, Ioannis Ch. Paschalidis, and Ayse K. Coskun. 2022. HPC Data Center Participation in Demand Response: An Adaptive Policy With QoS Assurance. *IEEE Transactions on Sustainable Computing* 7, 1 (2022), 157–171. https://doi.org/10.1109/TSUSC.2021.3077254

[30] Jiajia Zheng, Andrew A Chien, and Sangwon Suh. 2020. Mitigating curtailment and carbon emissions through load migration between data centers. *Joule* 4, 10 (2020), 2208–2222.