A Scalable Priority-Aware Approach to Managing Data Center Server Power

Yang Li[+‡] Charles R. Lefurgy[+] Karthick Rajamani[+] Malcolm S. Allen-Ware[+]
Guillermo J. Silva[+] Daniel D. Heimsoth[+] Saugata Ghose[‡] Onur Mutlu[*‡]

[+]*IBM* [‡]*Carnegie Mellon University* [*]*ETH Zürich*

ABSTRACT

Power management is a key component of modern data center design. Power managers must (1) ensure the cost- and energy-efficient utilization of the data center infrastructure, (2) maintain availability of the services provided by the center, and (3) address environmental concerns associated with the center's power consumption. While several power management techniques have been proposed and deployed in production data centers, there are still many challenges to comprehensive data center power management. This is particularly true in public cloud environments, where different jobs have different priority levels, and where high availability is critical.

One example of the challenges facing public cloud data centers involves power capping. As power delivery must be highly reliable and tolerate wide variation in the load drawn by the data center components, the power infrastructure (e.g., power supplies, circuit breakers, UPS) has high redundancy and overprovisioning. During normal operation (i.e., typical server power demands, and no failures in the center), the power infrastructure is significantly underutilized. Power capping is a common solution to reduce this underutilization, by allowing more servers to be added safely (i.e., without power shortfalls) to the existing power infrastructure, and throttling power consumption in the infrequent cases where the demanded power exceeds the provisioned power capacity to avoid shortfalls. However, state-of-the-art power capping solutions are (1) not directly applicable to the redundant power infrastructure used in highly-available data centers; and (2) oblivious to differing workload priorities across the entire center when power consumption needs to be throttled, which can unnecessarily slow down high-priority work.

To address this need, we develop CapMaestro, a new power management architecture with three key features for public cloud data centers. First, CapMaestro is designed to work with multiple power feeds (i.e., sources), and exploits server-level power capping to independently cap the load on each feed of a server. Second, CapMaestro uses a scalable, global priority-aware power capping approach, which accounts for power capacity at each level of the power distribution hierarchy. It exploits the underutilization of commonly-employed redundant power infrastructure at each level of the hierarchy to safely accommodate a much greater number of servers. Third, CapMaestro exploits stranded power (i.e., power budgets that are not utilized) in redundant power infrastructure to boost the performance of workloads in the data center. We add CapMaestro to a real cloud data center control plane, and demonstrate the effectiveness of all three key features. Using a large-scale data center simulation, we demonstrate that CapMaestro significantly and safely increases the number of servers for existing infrastructure. We also call out other key technical challenges the industry faces in data center power management.

1. INTRODUCTION

The power distribution infrastructure, which includes components such as power supplies, power feeds (i.e., sources), and circuit breakers, is a critical part of a data center, both in terms of its cost (tens of millions of US dollars) and its impact on availability. For *highly-available data centers*, the power distribution infrastructure often relies on redundancy at each level of the power distribution hierarchy to ensure reliable power delivery, spanning from multiple power supplies within individual servers all the way up to multiple utility feeds (i.e., external power sources) into the data center. This design, which uses *N+N redundancy,* provides two independent *sides* for the power distribution infrastructure, where each server is connected to both sides, and each side connects to an independent power feed. N+N redundancy ensures continued availability in the event of failure of the power devices on one complete side.

For a data center with N+N redundancy, a safe yet conservative design choice is to ensure that *each side* of the power distribution infrastructure on its own can power the peak (i.e., worst-case) power demands of *the entire center*. This approach ensures that a failure in the power delivery infrastructure does not lead to downtime, especially in cases where a data center's total power consumption exhibits wide variations, or cannot be anticipated in advance, such as in a public cloud. However, it also results in significant *overprovisioning* of power resources during normal operation, as the data center's total power consumption may be much lower than its peak power, resulting in underutilized infrastructure.

Many prior works [1, 3-5, 22-24, 28-31, 39, 40, 42-45] explore how to reduce overprovisioning for data centers *without* N+N redundancy. These works employ power capping, which throttles the amount of power consumed by servers during periods of peak demand, and, thus, reduces the load on the power distribution infrastructure. As a result, a data center can accommodate more servers for a given power distribution infrastructure than without power capping. With the gradual, industry-wide adoption of server power capping, today's data centers have the means to shape power consumption in real time, so that potential *power excursions* (i.e., cases where the total load may exceed the maximum power capacity of the infrastructure) can be avoided. However, in highly-available data centers with N+N redundancy, existing power capping based approaches are unable to reduce a significant amount of overprovisioning in the power distribution infrastructure, for three key reasons.

First, in highly-available data centers, a server draws power from multiple power supplies, each connected to a different power feed (i.e., source), to distribute the load of the server across both available feeds. We find that there is typically an imbalance in the power drawn from each supply. In such cases, power capping must ensure that the power consumed from *each* power supply does not exceed the supply's assigned power budget. Furthermore, the power load on one feed may be different from the load on the other (redundant) feed, due to the addition of non-redundant equipment or due to temporary power connectivity issues. This imbalance may require different power budgets for each power supply of the server. Unfortunately, state-of-the-art server power controllers [5-8] enforce only a *single* combined budget across *all power supplies*, and cannot ensure that the budgets for individual power supplies are respected. This can cause one of the power feeds to become overloaded, leading to tripped circuit breakers and power loss on the overloaded feed.

Second, existing power capping solutions are typically oblivious to the relative priority of each workload *globally* across the entire data center. State-of-the-art solutions such as Dynamo [5] are aware of workload priority, but only within a limited *local* group of servers. As a result of this limited view, existing techniques may unnecessarily cap a *critical* (i.e., high-priority) workload in one local group of servers, even though lower-priority workloads in another local group remain uncapped.

Third, existing power capping solutions cannot guarantee that the power budgets allocated to the individual power supplies of a server are fully utilized. This is because a server does *not* equally split its power load across its multiple power supplies, and the actual split is an intrinsic property of the server that cannot be adjusted at runtime. If the allocated power budgets do not match with this inherent power split, some of the power budgeted to one supply may not be fully utilized. This unutilized budget is known as *stranded power*.

To address these challenges, we propose *CapMaestro*, a new power capping architecture for public cloud data centers. CapMaestro unlocks the unused power capacity of a highly-available data center, which is provisioned for peak power and redundancy, to power more servers under a fixed power budget, while still protecting every level of the power infrastructure from overload (and, thus, avoiding data center downtime). CapMaestro has three key new features compared to state-of-the-art power capping architectures. First, it uses a new closed-loop feedback power controller for servers, which protects each individual power supply of a server from overload. Our new controller manages power consumption at each supply in response to the unique power loads and limits observed at each upper-level power infrastructure component, and therefore protects the safety of multi-feed, highly-available data centers. Second, CapMaestro performs efficient *global* priority-aware budget allocation using a distributed algorithm across multiple coordinated power controllers, which enables fault-tolerant and scalable power capping. This is significantly different from prior power capping solutions, which are either priority-unaware or only *locally* priority-aware. During a power emergency, CapMaestro shifts the *unnecessary* portion of power budgets (i.e., any power greater than the minimum needed) for low-priority workloads to high-priority workloads. As a result, a data center using CapMaestro can house more servers than a data center with state-of-the-art power capping, while still (1) protecting the high-priority workloads from being throttled and (2) guaranteeing the minimum required performance of low-priority workloads. Third, CapMaestro includes an optimization that adjusts the server power budgets to shift stranded power to servers that are currently throttled and, thus, improves server performance without exceeding global power budgets.

We implement CapMaestro as a scalable control plane service. Importantly, *our solution is designed to be applicable with minimal changes to existing data centers.* We use real-system experiments to demonstrate its effectiveness at power capping in a multi-feed data center power infrastructure with *global* priorities. To evaluate the effectiveness of CapMaestro, we perform a large-scale data center simulation using server load data for a Google data center [27]. We find that, for an example shared data center where 30% of the server workloads are high priority, CapMaestro enables the data center to support 50% more servers than if power capping was not employed, and supports 20% more servers than a state-of-the-art power capping solution [5] that we modified to support multiple power feeds. During a worst-case power emergency, we show that CapMaestro redistributes power such that the power emergency has only a negligible impact on the performance of high-priority workloads, compared to their performance during normal operation.

We make the following key innovations in CapMaestro:

- We show that servers with redundant power feeds do not often draw the same load from each power feed, and that this imbalanced load can negatively affect power budgeting during power capping.
- We propose CapMaestro, the *first* global priority-aware algorithm for data center power management that uses distributed, coordinated power controllers to enforce power limits at all levels of the power infrastructure.
- We design the *first* closed-loop feedback power controller for servers with *multiple* power supplies.

2. BACKGROUND

In this section, we first review a typical data center power infrastructure design, and then discuss power capping techniques at the server and data center levels.

2.1. Data Center Power Delivery

Figure 1 illustrates the power infrastructure for a typical data center. Power from the utility is delivered to the building at 12.5kV (❶ in Figure 1), and is stepped down to 480V for distribution. The 480V power then goes through an *automatic transfer switch* (ATS; ❷), which can switch to an on-site generator as a power source if the utility feed fails. Another layer of transformers (❸) after the ATS steps the voltage down further, providing a 3-phase line-to-line voltage of 400V. The corresponding line or phase voltage is 230V. *Remote Power Panels* (RPPs; ❹) are 42-pole boxes containing *circuit breakers* (CBs) that connect to *cabinet distribution units* (CDUs; ❺) in the racks. 3-phase power is delivered to the CDUs from

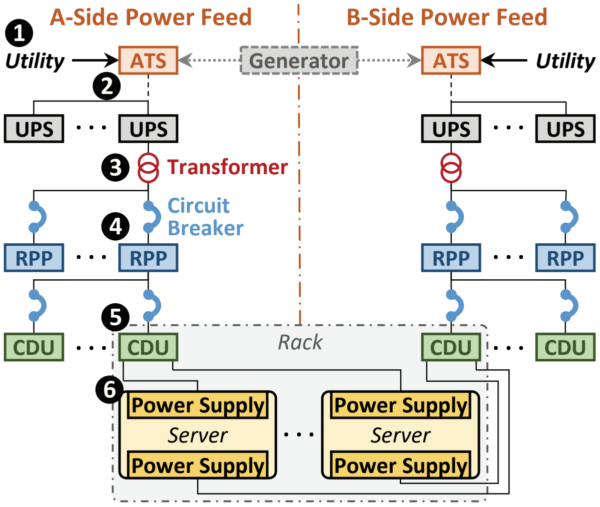


Figure 1. Example power distribution infrastructure in a data center.

the RPPs. The outlets on the CDU receive power at 230V from one of the three phases, and a server power supply (❻) is plugged into the outlet.

At each branch of a distribution point, there are CBs that limit the amount of current, to protect the power infrastructure and guard against cascading failures to upstream (i.e., higher) components due to short circuits and overload conditions. In this paper, we use the distribution point *power limit* to refer to the maximum amount of power that the corresponding CB or transformer allows. CBs are rated in terms of maximum current, but we convert them to their equivalent power values. When a CB trips, downstream power delivery is interrupted, potentially causing server power outage.

Redundant power feeds provide higher availability and resilience against power interruption. Servers rely on two or more power supplies connected to independent power feeds. Even if one of the power feeds or supplies fails, the remaining keep(s) the server operational. Ideally, power supplies should equally share the server power load, but in practice, the server load distribution varies from supply to supply.

Conventional practice in data centers is to not have sustained power load exceed 80% of the maximum rating for CBs and transformers [21] to avoid risk of damaging the power infrastructure. For example, a 30A 3-phase breaker may only be loaded to 24 A on each phase. When a power feed fails, its power load shifts to the remaining power feed, whose CB will see double the load. The server power connections must ensure that this doubled load doesn't exceed 80% of the CB rating. Otherwise, the CB may trip during the failure, and the servers downstream of the CB will lose power [5]. In our example with redundant (dual) 30A feeds, the per-phase load on each feed would need to be limited to 12A (40% of 30A) to ensure that the load during a failure is limited to 24A (80%).

In our work, we load CBs up to 80% under normal conditions because we employ power capping. When a feed fails, the breaker on the redundant feed becomes overloaded to 160%. The time it takes for a CB to trip depends on the amount of overload. For example, CBs covered under the UL 489 standard [17] (a widely-adopted industry CB standard) will operate for a minimum of 30 seconds before tripping when under 160% load [11, 17]. Within that 30-second window, our capping solution throttles the associated servers to ensure the load comes down to within 80% of the rating, which thus avoids tripping the CB and protects the servers against downtime.

2.2. Server Power Capping

In the past decade, power capping has become a common feature in servers to keep the server within a power limit [6, 7, 8, 25]. Typically, the power controller measures the server power and throttles the CPU (scaling its voltage and frequency) and other components to enforce the power limit. Prior work has shown that such controllers can generally perform decisions within a few seconds [6].

The range of power control can be characterized by running the most power-demanding workload at the highest and lowest performance states of the server at the highest allowed ambient temperature. For a server, P_{cap_min} is the power consumed by the server at the lowest performance state, and P_{cap_max} is the power at the highest performance state. Any power budgeted to the server above P_{cap_max} is wasted, as the server never uses the additional power, and a budget less than P_{cap_min} cannot be enforced by a power capping mechanism.

Capping server power consumption allows us to directly control the power load seen by CBs and transformers. The timescale of power capping is an order of magnitude faster than the trip time of CBs. This allows server power capping to adequately protect against tripping of CBs.

3. DESIGN GOALS

CapMaestro addresses three key challenges in leveraging server power capping for power management: (1) accounting for *multiple* power feeds (Section 3.1), (2) incorporating workload priorities *globally* in power capping decisions (Section 3.2), and (3) capturing and making use of *stranded power* in a redundant power infrastructure (Section 3.1). These challenges are important to address for highly-available data centers but have not been addressed in prior works.

3.1. Power Capping for Multiple Power Feeds and Stranded Power

The power load across multiple power feeds is not perfectly balanced. We observe this at all levels of the infrastructure for three reasons. First, servers with multiple power supplies do *not* split their power load equally between their power supplies. In our servers with two power supplies, there can be as much as a 15% mismatch across the two supplies, e.g., the power supply connected to the B-side feed can draw 65% of the total server power, as opposed to the expected 50%. This power mismatch varies from server to server and is an intrinsic property of each server (i.e., is independent of the workloads), and cannot be adjusted during use. Additional data is available in our technical report [41]. Second, power device failures (e.g., failed power supplies or power feeds) may also lead to imbalanced load between different power feeds. Third, to increase energy efficiency, some servers can put a redundant supply in standby mode (drawing no power) when the server load is below a certain level [34].

An imbalanced load between different power feeds forces power managers to assign and regulate a *separate* budget for each power supply of a server. For example, a server with two power supplies (Supplies A and B) may get a budget of 200W on Supply A (which is connected to the A-side feed) and only 100W on Supply B (which is connected to the B-side feed), because other servers exert a greater power load on the B-side feed, leaving less power available to assign to Supply B. However, existing server power capping solutions, which only limit the *combined load* across *all* power supplies, do not consider this need for individual per-supply power budgets, and therefore may not adequately protect the upstream circuit breaker for each power supply. To tackle this challenge, we propose a server power controller that enforces an individual power budget per supply in Section 4.2. Furthermore, in this example, when Supply B's power budget (100W) is fully consumed, Supply A's power budget (200W) may not be fully consumed, as these power budgets may not match with the way that the server's power load is split between its supplies. As a result, some portion of Supply A's budget would be left *stranded* (i.e. unutilized). It is desirable to reallocate (at least some portion of) stranded power to other power-constrained servers to improve their performance. To do this, we propose a stranded power optimization mechanism in Section 4.4.

3.2. Global Priority-Aware Power Capping

In a data center, some workloads can be more important than others (e.g., due to different pricing, workload heterogeneity, or service function). It is desirable to give sufficient power to these important workloads during power emergencies, so that their performance is not impacted. To do this, Facebook introduces the notion of *priority* in Dynamo [5], their power management system, to characterize the quality of power availability offered to different workloads. They assign higher priority to more important workloads, and try not to throttle them during a power emergency. This notion of priority is especially important in public clouds, where *service level agreements* (SLAs) are often expressed as the amount of resources (e.g., the number of CPU cores) that a user subscribes to, instead of a workload performance metric. This is because public cloud operators generally do not have access to user code, and are unable to measure performance directly.

One drawback of existing priority-based power allocation schemes, including Dynamo, is that to remain scalable, they consider priorities at best only between *local* groups of servers under a single power constraint (e.g., a branch circuit), and do not incorporate priority across higher levels of the power distribution infrastructure (e.g., they do not have a way to manage priorities across all of the servers connected to an RPP). This limits the ability to truly share power across the entire data center. In this work, we want to prioritize important workloads *globally* (across the entire data center) and give them sufficient power during a power emergency, by letting them borrow power from lower-priority workloads on a common power feed, regardless of the location of the physical server.

Figure 2 shows an example with four servers SA, SB, SC, and SD under a total power budget of 1240W for a single power feed. The servers connect to a power feed of three CBs: a top-level CB rated at 1400W, and two child CBs rated at 750W each, which we call Left CB and Right CB. These four servers have an equal power demand of 430W each, and they each have P_{cap_min} = 270W. Suppose SA has high priority, while the other servers have low priority (we use 2 priorities in this example for illustration; our mechanisms can support an arbitrary number of priorities). Table 1 shows how much power each server is budgeted under a *local* priority-aware power capping policy, and how power would be budgeted if the policy were instead *global* priority-aware. At the top level, the local priority-aware policy splits the total power budget equally across the Left and Right CBs, as only the lowest-level CBs have the knowledge of and enforce server priorities. Therefore, under a total power budget of 1240W, both Left CB and Right CB are assigned a power budget of 620W. As SB can at most be throttled down to 270W (P_{cap_min}), SA can only receive a power budget of 350W (i.e., 620W – 270W), even though it demanded 430W. In contrast, a global priority-aware policy knows *at the top level* that one of the servers under Left CB has high priority. As a result, the policy allocates more power to Left CB, allowing SA to be budgeted the full 430W that it demands. A global priority-aware policy ensures that a higher priority server is not throttled when lower priority servers *anywhere in the data center* can be capped to meet a given power constraint. We introduce a global priority-aware power capping algorithm in Section 4.3.

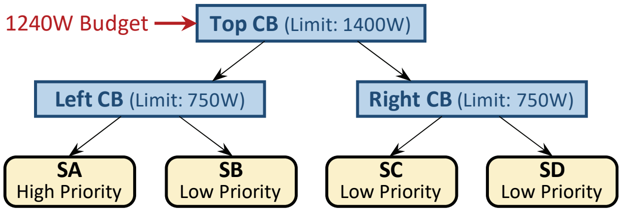


Figure 2. Example power feed with various server priorities.

Server	SA	SB	SC	SD
Priority (H = high, L = low)	H	L	L	L
Power Demand (W)	430	430	430	430
Budget with Local Priority (W)	350	270	310	310
Budget with Global Priority (W)	430	270	270	270

Table 1. Power budget assignments using local per-CB vs. global priorities.

4. CAPMAESTRO: DESIGN OVERVIEW

CapMaestro is a new scalable power management solution for data centers that achieves the three design goals described in Section 3. At a high level, CapMaestro employs a lightweight *power control framework* to efficiently collect power demand information and enforce power budgets for each node in the power infrastructure hierarchy, including each individual server power supply (Sections 4.1 and 4.2). CapMaestro uses the collected power demand information to determine power budgets for each node based on a new *global priority-aware power capping* algorithm (Section 4.3). Once global priority-aware allocation finishes, CapMaestro *optimizes stranded power* by identifying the nodes where assigned power is underutilized, and then reassigning this power where it is needed in the hierarchy (Section 4.4).

4.1. System Overview

CapMaestro uses a *power control tree* that mirrors the hierarchy of the power infrastructure. Figure 3 shows how the controllers in the tree map to the physical infrastructure components. At the bottom of the tree, *capping controllers* manage the power of individual server/IT equipment using the built-in server power capping mechanism. At each higher level of the tree, we use a power *shifting controller* that distributes the power budget at that level among the nodes fed from that distribution point. Each shifting controller is mapped to a single physical device, and adheres to (1) the device's *power limit* (e.g., maximum power allowed by a transformer, RPP, or CDU), and/or (2) a *contractual budget* (i.e., the maximum total power that a data center has negotiated to draw from the utility across all feeds).

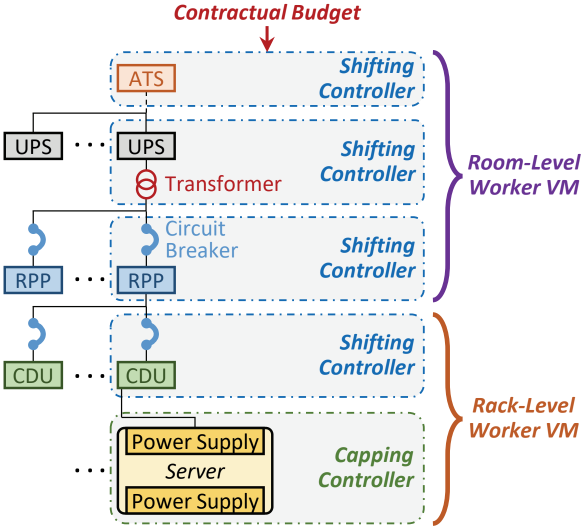


Figure 3. Mapping physical equipment from a single power feed to controllers in the power control tree, and to worker VMs.

To account for redundant power infrastructure, we replicate the power control tree for each power feed of the data center. We also replicate the power control tree for each phase of power delivery to protect each phase independently, since loading on each phase is not always uniform. For a data center with two power feeds with three phases each, our *power control framework* has six control trees. The shifting controllers on one power feed operate independently from shifting controllers on the other feed, while each server has a single capping controller that is shared across multiple trees. For a server with multiple power supplies, its capping controller adjusts the frequency/voltage of the entire server, affecting the load on *all* of the server's power supplies. This capping controller that addresses the power budgets for multiple supplies (from their corresponding trees) by controlling the server cap is a novel contribution.

Each server has a specific priority level. When servers run VMs or containers with different priorities, one could set server priority based on the priorities of the set of VMs/containers assigned to a server or assign VMs/containers to servers based on their priorities. A server's capping controller calculates relevant server metrics (e.g., power demand), which flow upstream in the control trees to the shifting controllers in the next level up. Each shifting controller produces priority-based metrics summarizing the sub-tree that it controls, based on the metrics that the shifting controller receives from its child nodes. *To perform global priority-aware power capping, a key insight is that we need to convey upstream only the metrics summarized by priority level, and not individual server metrics for all servers in a subtree*. In practice, we expect a data center to have only a small number of priority levels (on the order of 10); thus, the priority-based summaries provide us with a compact way to represent metrics for thousands of servers. This allows the shifting controller at the root node to efficiently have a global view of the power demand across the entire data center. With this view, the root shifting controller easily routes power (by assigning power budgets to its child nodes) towards the most critical servers by comparing priority-based metrics from each of its child nodes, while respecting the power limits of the intervening CBs and transformers along the control tree. These budgets flow downstream, and are recursively allocated until the budgets reach the capping controllers (see Sections 4.3 and 4.4 for algorithm details). After a power budget is assigned to a capping controller, the controller (Section 4.2) ensures that for *each* power supply of the server, the per-supply power budget is not exceeded by the power consumption on that supply.

Our control trees mirror the physical electrical connections of the data center, allowing us to model situations unique to each data center or portions of it. For example, CapMaestro can (1) manage both multiple- and single-corded devices; (2) deal with equipment that does not include power capping technology, by setting the metrics to assign a fixed maximum power for that equipment; (3) capture servers plugged into multiple phases of power; and (4) work with power budgets based on restrictions aside from physical equipment limits, e.g., contractual budgets.

4.2. Power Supply Budget Enforcement

To protect the independent power feeds of the redundant power infrastructure, we design a proportional-integral (PI) [15] feedback controller for CapMaestro that guarantees adherence to AC power budgets on the power consumption of each power supply in a server. Our controller employs the server power capping controls of Intel Node Manager [7], which cap only the total DC power of the server. The input to our controller is the external AC power budget for each power supply. These budgets are determined by the power capping algorithm that protects each power feed. The budgets for the power supplies of a server may have unequal values, depending on the load on each power feed. The controller determines the proper DC power cap for the node manager to enforce the AC power budgets for all supplies.

Figure 4 shows our control diagram. First, each control iteration calculates an *error* for each power supply by subtracting its measured power from its budget (❶ in Figure 4). This error quantifies how much the drawn AC power falls short of the assigned AC budget on each power supply. Then, the controller selects the minimum error across all power supplies (to make the most conservative correction).

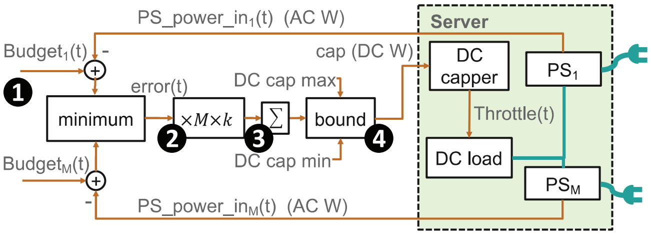


Figure 4. Capping controller enforcing budgets on power supplies (PS).

The stranded power optimization mechanism will later shift the unused power budgets to other servers for better utilization (Section 4.4). Second, the minimum power supply error is scaled by the power supply efficiency (k) to transform from AC power domain to DC power domain (k can be determined from the power supply specification), and then further scaled by the number of working (non-faulty) power supplies (M) to account for how much DC power the full system power needs to be adjusted by (❷). Third, the scaled error is added to the integrator (which stores the previously desired DC cap) to calculate the currently desired DC power cap (❸). The controller then clips the calculated DC power cap based on the maximum and minimum power values that the controller can cap to (❹), and sends the cap value to the node manager, which manages the processor frequency and voltage to meet the DC power cap.

4.3. Global Priority-Aware Power Capping

CapMaestro's global priority-aware power capping algorithm allocates power budgets across a control tree of shifting and capping controllers, respecting the data center contractual budgets and the power limits of multiple levels of CBs and transformers while safely trying to satisfy as much of the power demand of each server. For both non-redundant and N+N redundant power distribution infrastructures, each control tree runs this algorithm independently.

Our algorithm runs iteratively, with each iteration consisting of two phases. In the **metrics gathering phase**, each shifting controller receives power allocation requests (and other metrics) from its child nodes. These metrics are grouped by priority value, which corresponds to the priority level of the workloads running on the servers under the child node. The shifting controller then aggregates these metrics from all its children by priority value, and sends the aggregated metrics upstream to its parent node. In the **budgeting phase**, each shifting controller receives its power budget from its parent node, and then computes and sends power budgets downstream for its child nodes based on the power budget assigned to the controller and the priority-based metrics of its child nodes. At the bottom, each capping controller receives an individual budget for each of its power supplies (from the corresponding *leaf* shifting controller), and uses the method discussed in Section 4.2 to set a power cap for the corresponding server.

4.3.1. Metrics Gathering Phase

CapMaestro computes the following metrics at each *node* (a node may correspond to a shifting or capping controller) at level i of the power distribution hierarchy (where level 0 corresponds to a server, level 1 corresponds to the per-server capping controllers, and higher level numbers correspond to shifting controllers):

- $P_{cap\_min}(i, j)$: the minimum total power budget that must be allocated to servers with priority level j under the node.
- $P_{demand}(i, j)$: the total power demand of servers with priority level j under the node.
- $P_{request}(i, j)$: the power budget that is requested by servers with priority level j under the node. If the node corresponds to a capping controller, this will be the power demand of the single server governed by the capping controller. If the node corresponds to a shifting controller, this may be lower than the total power demand of servers with priority level j under the node, because the servers may demand more power than what the circuit breakers under the node allow.
- $P_{constraint}(i)$: the maximum power budget that can be allocated safely to *all* servers under the node (no matter what their priorities are). This metric is limited by the power limit of the node, power limits for downstream shifting controllers, and P_{cap_max} for the downstream capping controllers.

The computation of metrics differs between the capping and shifting controllers. At each ***capping controller***, when j equals the priority level of the server whose power supply is being governed by the controller, we calculate the metrics as:

$$P_{cap\_min}(1, j) = r \times P_{cap\_min}(0)$$

$$P_{demand}(1, j) = r \times \max\{P_{demand}(0),\ P_{cap\_min}(0)\}$$

$$P_{request}(j) = P_{demand}(j)$$

$$P_{constraint} = r \times P_{cap\_max}(0)$$

where r is the fraction of the server load borne by that power supply (nominally $1/M$, where M is the number of working power supplies; we adjust it in practice based on how the load is actually split between the working power supplies of the server), $P_{cap\_min}(0)$ and $P_{cap\_max}(0)$ are the minimum and maximum controllable AC power budgets for the server, and $P_{demand}(0)$ is the amount of power that workloads running on the server consume at full performance (we discuss how to estimate this in Section 5). Since the capping controller governs only one server power supply, its minimum power budget depends solely on the minimum power budget for the server. When we calculate $P_{demand}(1, j)$, we choose the maximum of $P_{demand}(0)$ and $P_{cap\_min}(0)$, and then scale it with r. This is because if the server is running light workloads, $P_{demand}(0)$ may be less than $P_{cap\_min}(0)$. In this scenario, our power capping algorithm needs to ensure that the aggregate power budget allocated to the server across its power supplies stays within the controllable range; otherwise, the power cap on the server may not be enforceable if the server load suddenly increases later. For j not equal to the server priority, the corresponding metric values are zero.

At each ***shifting controller***, we calculate the metrics, in descending order of priority (i.e., highest priority first), as:

$$P_{cap\_min}(i, j) = \ \Sigma_k P_{cap\_min_k}(i-1, j)$$

$$P_{demand}(i, j) = \ \Sigma_k P_{demand_k}(i-1, j)$$

$$P_{request}(i,j) = \min \begin{Bmatrix} P_{limit} - \sum_{h>j} P_{request}(i,h) - \sum_{l<j} P_{cap\_min}(i,l), \\ \Sigma_k \, P_{request_k}(i-1,j) \end{Bmatrix}$$

$$P_{constraint}(i) = \min\{P_{limit}, \ \Sigma_k P_{constraint_k}(i-1)\}$$

where k is a child node iterator; j, h, and l are priority levels; and P_{limit} is the power limit of the shifting controller. For P_{cap_min} and P_{demand}, we aggregate the corresponding metrics that each of the child nodes k on level i–1 report to the shifting controller. To calculate $P_{request}$, we need to consider two factors. First, for a priority level j, no server with a higher priority level h (i.e., $h > j$) should be capped before a server at priority level j. Therefore, servers at priority level j are allowed to request as much power as they can, provided that all *higher-priority* servers get *all* of the power that they request, and that all *lower-priority* servers have enough power left over to operate at their *minimum* possible power level. We call this the *maximum allowable power request* for priority level j. Second, if the maximum allowable power request is greater than the actual sum of power requested by the child nodes at priority level j, then the total power request for level j is reduced to the actual requested power. We set $P_{constraint}$ to the lower of (1) the power limit of the current shifting controller and (2) the sum of the power limits of the controllers at level i–1.

4.3.2. Budgeting Phase

The budgeting phase at each shifting controller distributes its power budget among its child nodes in four steps:

1. Allocate P_{cap_min} of power to each child.
2. Iterate over the priority levels (j) from highest priority to lowest priority, to further allocate any *additional* power requested by each child node (i.e., $P_{request} - P_{cap\_min}$). If the power remaining in the controller's budget is not enough to meet the power requested for any priority level during this step, go to Step 3; else, go to Step 4 after finishing *all* priority levels.
3. For the last priority level j whose power demand could not be completely fulfilled in Step 2, proportionally give the remaining budget to each child k based on the amount of power that the child demands over its minimum required power ($P_{demand} - P_{cap\_min}$).
4. If there is any unallocated power remaining after fulfilling all power requests, assign this power to the child nodes up to $P_{constraint}$.

We have rigorously proven that our global priority-aware power capping algorithm allows servers with high priority to always be throttled after servers with lower priorities, as long as the power limits in the data center allow. The proof is available in our extended technical report [41].

4.4. Stranded Power Optimization

As we discuss in Section 3.1, power loads may be imbalanced between different power feeds of a data center. This can lead to mismatched power budgets for a server's power supplies, such that a portion of the power budget assigned to one of the supplies may be *stranded* (i.e., unutilized). Ideally, a power capping algorithm should reassign this stranded power to the budget of another server on the same power feed that requested more power than it has currently been budgeted.

Our stranded power optimization (SPO) mechanism runs after CapMaestro performs the global priority-aware power capping algorithm. With SPO, CapMaestro does not apply the budgets generated by power capping immediately. Instead, based on each power supply's assigned budget, and the inherent distribution of load between the multiple power supplies of a server, CapMaestro determines which supplies have stranded power, and then reduces the power budgeted to each of these supplies to the actual amount that the supply can use (i.e., such that no stranded power remains). This frees up the stranded power for re-budgeting elsewhere in the feed. Once this requested power reduction is complete for all supplies, CapMaestro runs the power capping algorithm a second time, which shifts the previously-stranded power (now unallocated) to servers that were capped before SPO.

5. IMPLEMENTING CAPMAESTRO

We implement a prototype of CapMaestro as an integral service that can be run in a *real* cloud data center control plane. We group and run the shifting and capping controllers of CapMaestro into VMs called *worker VMs*, as shown in Figure 3. Our system is flexible in terms of (1) the mapping of controllers to worker VMs, and (2) supporting an arbitrary arrangement of a multi-level worker hierarchy. A good mapping should be based on (1) the number of servers deployed and (2) the configuration of the power distribution hierarchy in the data center.

For a typical multi-rack data center, we envision that the data center manager would deploy VMs for (1) rack-level workers for each rack of servers, and (2) a room-level worker for the entire data center. Each *rack-level worker* protects its assigned rack's CDU (cabinet distribution unit; see Section 2.1). One rack-level worker contains 6 CDU-level shifting controllers (one for each phase, where we have 2 feeds, and 3 phases per feed) and 45 *capping controllers* (one capping controller for each server in the rack). The worker calculates the server metrics discussed in Section 4.3 (e.g., power demand, minimum power budget, requested power budget), communicates with the upstream worker to exchange metrics and receive budgets, and assigns power budgets to the server's node manager every control period (8 seconds in our setup). The *room-level worker* protects RPPs (redundant power panels; see Section 2.1), transformers, and the contractual power budget that governs all racks. Like the rack-level workers, the room-level worker calculates metrics and determines budgets every control period (8 seconds).

For our real-system experiments in Section 6, we deploy our CapMaestro prototype across a small test bed with a data center control plane. In this case, we use a single worker that consists of one capping controller per server, and two levels of shifting controllers. The controllers in the single worker faithfully execute the entire CapMaestro algorithm.

Every second, each capping controller reads sensors for the server under its control, using the Intelligent Platform Management Interface (IPMI) [26]. The sensors include AC power monitors for the two power supplies and the *power cap throttling* level. The power cap throttling level is an Intel Node Manager [7] metric that quantifies the current fraction of server voltage/frequency throttling. Every 8-second control period, the capping controller averages the per-second readings and computes the server-level metrics. By averaging the readings at this granularity, CapMaestro's power capping controller provides a more stabilized response while still being fast enough to address failures in the power distribution infrastructure. The capping controller then sends the metrics to upper-level nodes, and shortly thereafter receives a budget to allocate to each power supply. The capping controller sets the DC power cap based on this budget by sending the cap to the node manager on the server's baseboard management controller [33] via IPMI [26]. The node manager then ensures that the server power is within the cap in 6 seconds. In other words, a new power cap is set in at most 14 seconds, well within the 30-second window during which the infrastructure components can tolerate exceeding the power limit if a failure occurs (see Section 2.1).

Each capping controller computes the power demand (P_{demand}) of its assigned server using a regression method [16]. The capping controller uses per-second readings of the server power consumption and power cap throttling level over the last 16 seconds, and builds a regression model that correlates server power to throttling levels. With this model, the controller can estimate the server power at 0% throttling (i.e., the maximum power consumed by the server for the workload), which it sets as P_{demand}. If power is measured during an interval when the power cap throttling is set to 0%, then the controller uses the actual measured power instead of the regression model prediction.

Overhead and Scalability Analysis. CapMaestro has a minimal cost and good scalability. First, our control framework makes use of existing node managers, such as the Intel Node Manager [7], to control server power, which are already built into the firmware of each server. CapMaestro requires no additional overhead to make use of the existing node manager. Second, one level up from the servers, we reserve one core per rack to run a rack-level worker (each rack has 1260 cores; therefore, 1/1260 = < 0.1% overhead), which completes rack sensing in 1 second (sensing all nodes in parallel), rack budgeting in 10 ms, and capping (i.e., calculating and sending the DC power cap to each server) within 1 second. The communication between this worker and the upstream worker is on the order of milliseconds, as we measure in real systems. The computation and communication overhead of a rack-level worker does not change when we add more racks for large data centers, allowing the worker to scale easily to large data centers. Third, at the top of our control framework, we reserve one core in the data center to run the room-level worker. The computation time of a shifting controller grows linearly with the number of its child controllers. Given that, and the fact that the room-level worker has no capping controllers, we estimate the computation time for the room-level worker deployed on a large data center with 500 racks to be well under 300 milliseconds (500/45 x 10 milliseconds x 2 feeds < 300 milliseconds). Hence, the room-level worker scales to realistic data center sizes. In summary, CapMaestro uses less than 0.1% of the data center's resources, and can scale well for large data centers.

6. EVALUATION

We experimentally evaluate each aspect of CapMaestro on a real small-scale cloud test bed, demonstrating that it can successfully (1) enforce different budgets for multiple power supplies of a server using server power capping (Section 6.1), (2) perform global priority-aware power capping across hierarchical power constraints in the entire data center (Section 6.2), and (3) optimize stranded power for redundant power feeds (Section 6.3) based on real system experiments. Our test bed contains 28-core servers that each run the Apache HTTP Server [18] as a representative cloud workload (with a separate client cluster running the Apache benchmarking tool *ab* [19], which is not part of our evaluation). Aside from our real system experiments, we determine the improvement in server capacity under CapMaestro by performing a large-scale data center simulation based on characteristics measured from our real servers (Section 6.4).

6.1. Server Power Cap Enforcement

Figure 5 shows how our controller in Section 4.2 enforces power budgets on the individual power supplies (labeled PS1 and PS2 in the figure) of a representative server from our test bed. We make two observations from the figure. First, at the beginning of the execution period, the budgets for both supplies are higher than the loads, and there is no throttling, as each supply is budgeted more than enough power. Second, at t=30s, when we lower the budget for PS2 to 200W, the CapMaestro controller prototype responds by computing and applying the resulting DC cap for the server that lowers PS2's power down to the new budget. The node manager then applies the DC cap to the server, which lowers the server load on *both* PS1 and PS2. Third, at t=110s, when we assign an even smaller budget of 150W to PS1 (making it the more constrained of the two power supplies), CapMaestro computes and applies the corresponding DC cap to bring down the server load, which reduces both supplies' power consumption. In both cases, CapMaestro's power capping controller recognizes which of the power supplies has the more constrained budget, and ensures that the server load is lowered enough so that the power supply loads satisfy the more

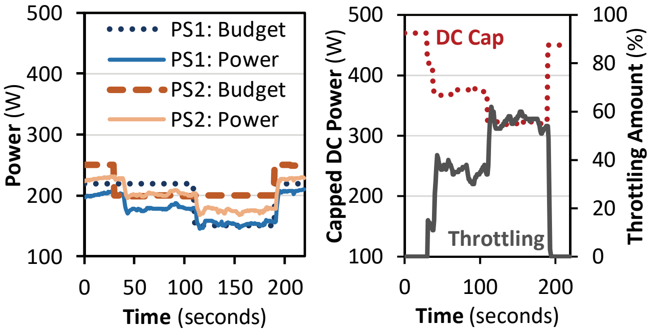


Figure 5. Power capping for redundant power supplies (PS1 and PS2). *Throttling* refers to power cap throttling.

constrained budget. Overall, the power settles to within 5% of the assigned budgets within two control periods (16 seconds).

6.2. Comparison of Power Capping Policies

To evaluate the benefits of global priority-aware power capping, we set up four real servers using the hierarchy shown in Figure 2. We run experiments on these servers to represent the conceptual example that we describe in Section 3.2, where we need to budget power to a combination of high-priority and low-priority servers. In our setup, all four servers are powered by a single power feed, to emulate a power failure scenario in a redundant power infrastructure where a second power feed has failed (which necessitates power capping). Each server consumes an average of 420W without power capping, and has a minimum power consumption (P_{cap_min}) of 270W. Server SA is assigned high priority, and the other three servers (SB, SC, and SD) are assigned low priority.

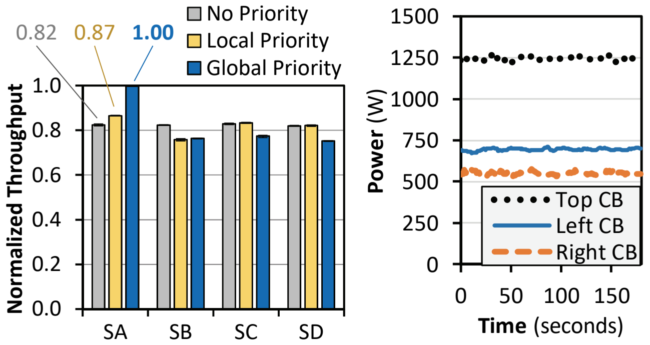
We evaluate the power allocated to each server under three different power capping polices: *No Priority*, *Local Priority* (a version of the state-of-the-art Dynamo [5] mechanism that supports redundant power feeds), and *Global Priority* (i.e., our policy under CapMaestro). A *No Priority* power capping policy, after guaranteeing that each server receives at least P_{cap_min}, distributes the remaining power proportionally to each server based on the server's value of $P_{demand} - P_{cap\_min}$. Our version of the *Local Priority* capping policy enforces the notion of priority only at the lowest controller level, while the higher-level controllers distribute power to each branch using a No Priority policy. To implement the Local Priority policy, we extend Facebook's Dynamo [5] to support power budget assignments and capping for a redundant power infrastructure. Our *Global Priority* policy in CapMaestro enforces a common priority system at every power controller (Section 4.3). For all three policies, we set the total power budget of the servers to 1240W. Since this does not cover the full 1680W demanded by all four of the servers, each policy needs to perform some form of power capping.

Table 2 shows the power budgets assigned by these three policies to each of the four servers (SA, SB, SC, and SD) when the workloads are in a steady state. We observe that our Global Priority policy for CapMaestro lets the high-priority server SA consume 419W, which is very close to its full power demand (420W). In contrast, the Local Priority and No Priority policies cannot allocate the full power to SA despite its high priority. In particular, the Local Priority policy recognizes priorities in only a local group, and so it can allocate power from SB to SA to alleviate the capping of SA's workload, but it cannot allocate power from SC and SD. The Global Priority policy can redistribute power to SA from all three other servers. This allows the workload running on SA to achieve a higher throughput and lower latency with Global Priority than with the Local Priority or No Priority policies.

Figure 6a shows the measured throughput of each server, normalized to an ideal baseline where no power capping takes place (where the throughput is defined as the number of queries completed per second). We make two observations from the figure. First, for SA, the No Priority policy results in 18% lower throughput (and 21% higher average latency) relative to the uncapped performance of SA, while the Local Priority policy results in 13% lower throughput (and 15% higher latency). With the Global Priority policy, SA achieves the same throughput (and latency) as if it were uncapped. Second, the improved throughout for SA under Global Priority comes with only a small reduction in throughput for the other three servers. We conclude that Global Priority is an effective method to maximize the throughput of higher-priority jobs that may exist *anywhere* in the data center.

Server	**SA**	**SB**	**SC**	**SD**
Priority (H = high, L = low)	H	L	L	L
Power Demand (W)	420	413	417	423
Budget with No Priority (W)	314	306	311	316
Budget with Local Priority (W)	344	274	314	317
Budget with Global Priority (W)	419	276	275	275

Table 2. Server power budgeted by each power capping policy.



(a) Server throughput after capping

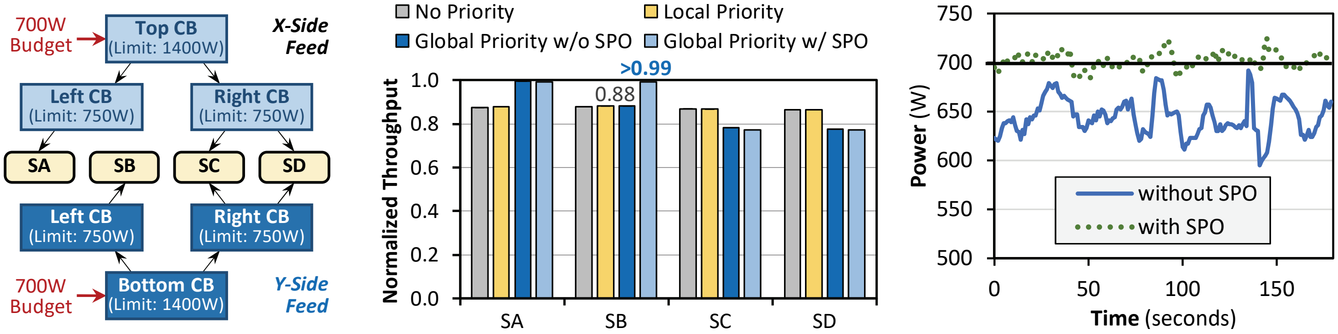
(b) CB power

Figure 6. (a) Server throughput after power capping policies, normalized to uncapped server throughput; (b) Power at each CB under Global Priority.

Figure 6b shows the total power consumption that we measure at the top, left, and right circuit breakers (CBs) under our Global Priority policy. We observe that the total power consumption is below the respective limits of the top CB (1240W), and of the left and right CBs (750W). This demonstrates that our policy can successfully redistribute power while ensuring that the actual power consumption respects power limits and budgets in the data center, which in turn guarantees power safety.

6.3. Impact of Stranded Power Optimization

To demonstrate CapMaestro's Stranded Power Optimization (SPO), we connect four servers (SA, SB, SC, and SD) to two redundant power feeds (X-side and Y-side), as shown in Figure 7a. Server SA has high priority, while the other three servers have low priority. We disconnect the Y-side supply of SA and the X-side supply of SB. SC and SD draw power from both power feeds (albeit unequally due to the inherent power split mismatch of each power supply belonging to a server). Each power feed is given a budget of 700W (i.e., the total budget is 1400W), and the rating for the top and bottom CBs is set at 1400W each in order to provide N+N power delivery in the event of a power failure. We set the rating of the left and right CBs to 750W each.



(a) Power feed organization (b) Server throughput normalized to uncapped throughput (c) Power used by the Y-side feed

Figure 7. Stranded power optimization evaluation.

Server		**SA**	**SB**	**SC**	**SD**
Priority (H = high, L = low)		H	L	L	L
Demand (W)		414	415	433	439
Global Priority w/o SPO (W)	Budget	415/0	0/346	152/***164***	132/***187***
	Consumption	413/0	0/348	156/***137***	135/***158***
Global Priority w/ SPO (W)	Budget	416/0	0/413	152/132	132/155
	Consumption	413/0	0/412	153/134	133/156

Table 3. Server power budgets and actual power consumption for our stranded power studies (power listed as X-side/Y-side). *Red italics* indicate that a server has stranded power (i.e., more power budgeted than consumed).

Table 3 shows the allocated power budgets and the average power measured at each server on the X-side and Y-side power feeds, under different power capping policies. For Global Priority power capping without SPO, SC and SD receive a power budget of 164W and 187W, respectively, on the Y-side. However, SC and SD actually use only 137W and 158W, respectively, from the Y-side, due to the lower amount of power budgeted by the X-side feed, which brings down total power consumption. This leaves 27W and 29W stranded on the Y-side feed for SC and SD, respectively. If we apply CapMaestro's SPO mechanism (i.e., Global Priority w/ SPO), the mechanism lowers SC's and SD's Y-side power budgets to 132W and 155W to match the actual power consumed by the servers, and SPO shifts 67W of underutilized power to SB to reduce the amount of capping needed on SB.

Figure 7b shows the throughput of the four servers without and with SPO. We make two observations from the figure. First, as a result of the redistribution of stranded power under SPO, SB approaches its uncapped throughput, as shown in Figure 7b. In contrast, without SPO, SB has a 12% lower throughput (and 14% higher latency) relative to its uncapped performance. As we see in Figure 7c, with SPO, the Y-side feed of the data center consistently uses the full budgeted power throughput the entire execution time. This additional power usage allows SB to maintain the higher throughput shown in Figure 7b. Second, after SPO runs, the throughputs of SC and SD remain unchanged from the throughput before SPO. This confirms that the power that SPO identified as stranded was not being used by the servers. We conclude that SPO is an effective mechanism to redistribute power stranded by some servers in a way that boosts the throughput of other servers in a data center.

6.4. Data Center Capacity Improvement

We perform large-scale simulations to study the number of servers that a data center can support when CapMaestro is employed under different conditions, compared to state-of-the-art power management policies. We consider both (1) *typical-case* conditions, where the servers experience normal load and where both feeds in the power infrastructure are fully operational; and (2) *worst-case* conditions, where all servers request maximum power and one entire power feed is down.

Data Center Configuration. Our simulations model a production data center infrastructure (shown in Figure 1), using the parameters summarized in Table 4. The data center has 2 three-phase power feeds (X-side and Y-side), 4 transformers, 36 RPPs, and 324 CDUs, for a total of 162 racks (two CDUs, one from each feed, power one rack). We designate 30% of the servers as high-priority, selecting the servers at random throughout the data center (we perform sensitivity studies on the fraction of high-priority servers in our technical report [41]). We load the circuit breakers and transformers to 80% of their maximum rated power. The contractual budget for the data center is 700kW per phase, or 2.1MW in total. We use 95% loading for this contractual budget, and reserve 5% as a margin to tolerate errors (e.g., server parameter error, power measurement error). Without employing a power management system, each phase of the CDU can serve at most 8 servers (700kW x 95% / 162 CDUs / 490W per server = 8.4 servers) at peak power demand, resulting in a total of 3888 servers deployed in the data center.

Contractual Budget	700kW per phase, split over two feeds
Transformers	2 per feed, rated at 420kW each
Remote Power Panels (RPPs)	9 per transformer, rated at 52kW each
Cabinet Distribution Units	9 per RPP, rated at 6.9kW each
Servers	6–45 per rack; idle power = 160W, P_{cap_min} = 270W, P_{cap_max} = 490W

Table 4. Simulated data center parameters.

Load. For typical-case conditions, we use a load profile released by Google [27] as our *typical load*. This load profile, shown in Figure 8, contains the distribution of average CPU utilization in a shared data center over time. For worst-case conditions, all servers have 100% CPU utilization.

Simulation Methodology. In our simulations, we vary the total number of servers deployed in the data center by

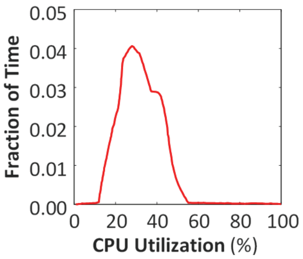


Figure 8. Distribution of average CPU utilization [27].

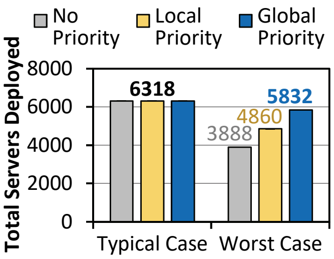


Figure 9. Total servers deployable (30% of servers are high-priority).

changing the number of servers in each rack (from 6 to 45), while keeping the rest of the infrastructure constant. This allows us to study how the workloads are capped under a different total number of servers. For typical-case simulations, we perform 20k Monte Carlo simulations for each server count, per power management policy evaluated. For each typical-case simulation, we select an average CPU utilization for all servers in the data center from the distribution in Figure 8, and vary the CPU utilization of each server randomly around the average value using a normal distribution. For worst-case simulations, we perform 1k Monte Carlo simulations for each server count, as we find that our results converge much earlier for worst-case conditions due to the constant CPU utilization.

We determine each server's power demand (which must fall somewhere between its idle power and maximum power) using the server's assigned CPU utilization. We calculate how CPU utilization correlates to power consumption using a power model from prior work [2]. We allocate a power budget to each server using one of the three power management policies discussed in Section 6.1: No Priority, Local Priority, and Global Priority. Our simulation selects which servers are high-priority at random for each simulation.

Metrics. To quantify how power capping affects performance, we define a metric that we call the *cap ratio*, which is the fraction of a server's dynamic power demand (i.e., non-idle power) that is capped by the assigned budget:

$$Cap\ Ratio = \frac{Demand - Budgeted\ Power}{Demand - Server\ Idle\ Power}$$

The cap ratio provides us with an application-neutral way of characterizing the *maximum* impact on performance imposed by capping (the actual impact will be lower, as power consumption is linear or superlinear with performance). A *lower* cap ratio is *better*.

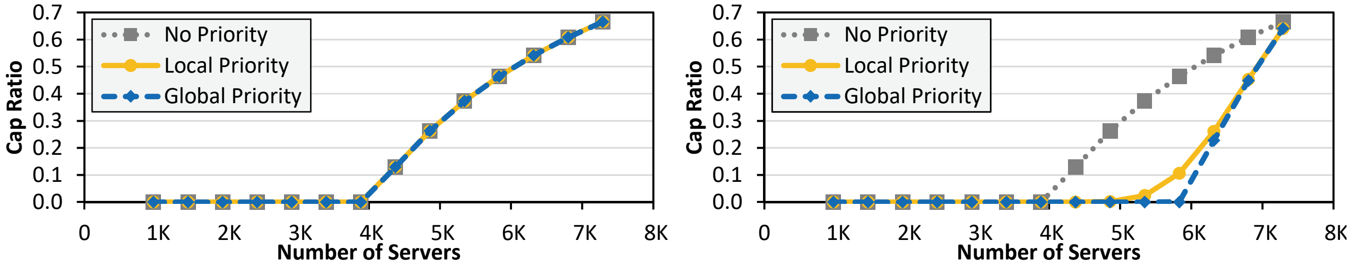
Results. Figure 9 shows the maximum number of servers that can be deployed under each of the three evaluated power management policies, under both typical-case and worst-case conditions, for a data center set up using the parameters in Table 4. Our goal is to increase server count while negligibly impacting both (1) the average performance of *all servers during typical-case conditions*, and (2) the average performance of *high-priority servers during worst-case conditions*. We consider anything less than a 1% average cap ratio across the servers to be a negligible performance impact. We make three observations from the figure.

First, under typical-case conditions, all three policies can support the same maximum number of servers (6318 servers in total). This is because while the three policies differ in terms of how they handle high-priority servers, our cap ratio criterion for typical-case conditions does not differentiate between high-priority and low-priority servers. In the typical case, a data center should ideally be able to avoid capping most, if not all, of its servers.

Second, under worst-case conditions, CapMaestro's Global Priority policy supports 50% more servers than the No Priority policy, and 20% more servers than Local Priority. During worst-case conditions, a significant amount of power capping needs to take place to bring the power load to within the rated limits of each component of the power distribution infrastructure. Both Dynamo and CapMaestro work to minimize the impact that this capping has on high-priority servers, while still allowing low-priority servers to make forward progress. However, CapMaestro has the ability to make *global* power capping decisions, and can redistribute power efficiently across multiple levels of the power distribution hierarchy. As a result, CapMaestro's Global Priority policy can support a much greater number of machines than the Local Priority policy based on Dynamo [5].

Third, CapMaestro's Global Priority policy can maintain most of the servers that a failure-free power infrastructure can support. In an ideal case where the power delivery infrastructure guarantees that no one component would fail, the data center could support all 6318 servers possible under typical-case conditions. The Global Priority policy can support 92.3% of the total ideal server count, by supporting up to 5832 servers. In contrast, the No Priority and Local Priority policies support only 61.5% and 76.9% of the total ideal server count.

Figure 10 shows how the average cap ratio changes under each of the three evaluated power management policies as we increase the number of servers. The x-axis shows the number of servers, and the y-axis shows the corresponding cap ratios



(a) Cap ratio for all servers

(b) Cap ratio for high-priority servers

Figure 10. Average cap ratio for all servers and high-priority servers during a worst-case power emergency.

for all servers (Figure 10a) and for high-priority servers (Figure 10b), under a worst-case power emergency (i.e., an entire side of the power infrastructure has failed). We make two observations from the figure. First, the cap ratios grow with the number of servers deployed in the data center. For the priority-aware policies, high-priority servers are throttled last, only after all low-priority servers are throttled. Therefore, the high-priority servers have a lower cap ratios under these policies compared to the cap ratio for all servers. Second, as we see in Figure 10b, high-priority servers perform better under Global Priority than under Local Priority. This is because Global Priority lets high-priority servers take power from low-priority servers even when the servers fall under *different* shifting controllers. In contrast, the Local Priority policy can only redistribute power among servers under the same shifting controller.

We conclude that CapMaestro, with its global priority-aware power capping, effectively supports a much greater number of servers for a given power distribution infrastructure. Thus, CapMaestro significantly reduces the impact of a failure in the power distribution infrastructure compared to state-of-the-art mechanisms.

Sensitivity Studies. We perform several sensitivity studies on how CapMaestro performs as we change key data center parameters, including (1) the fraction of servers that are high-priority, (2) the value of P_{cap_min} (i.e., the minimum power that can be budgeted to each server), and (3) the contractual budget of the data center. We find that Global Priority outperforms Local Priority and No Priority under most scenarios. Due to space limitations, we provide details on these studies in our extended technical report [41].

7. DISCUSSION: OPEN CHALLENGES

Aside from the challenges that we successfully tackle in this work, there are a number of open challenges that remain against the comprehensive adoption of power management for public cloud data centers. We provide a brief discussion of these challenges, with the hope of motivating future work in these and other related areas.

Limited Availability of Power Capping. Existing power capping controllers, such as the Intel Node Manager [7] and RAPL [46], control server power only through processor and memory throttling. In a data center, we ideally want to monitor and control the *total system power*, including storage devices, networking, and coprocessors such as GPUs and FPGAs. These components may consume significant amounts of power in contemporary systems. While some prior works [12, 13] study the power behavior of individual components in data centers, a *comprehensive* integration of power control for acceleration components into a server power controller does not yet exist. As part of a comprehensive solution, there is a need to provide dynamic power control for storage and networking equipment, as these components have at best only limited control today.

Specification and Standardization Gaps. We find that many available power measurement and power capping tools do not provide specifications for measurement and control accuracy, and do not guarantee responsiveness. Despite the lack of available information, these specifications are important for data center designers to take into account, as the specifications affect the margins that designers need to allocate to each power management component in the data center. A related issue is the limited standardization of power control interfaces across vendors. This can make the complexity of interfacing and controlling data center equipment from different vendors intractable.

Limited Emphasis on Power Infrastructure Topology. The physical topology of the power infrastructure is important, as it poses unique constraints on power management solutions. Unfortunately, this topology is neglected by many works. The physical topology of the power infrastructure is an essential factor in the performance of CapMaestro (e.g., we are the first to consider the redundant feeds in highly-available data centers). For future work, we observe that there are no common tools for expressing the physical power topology, or for validating a power topology at runtime. For example, wiring mistakes are possible when we connect servers to the power infrastructure (e.g., a wire is not plugged into the correct outlet). There is a need to develop a cost-effective approach to finding such errors in the topology (other than manual cable tracing), or for tolerating such power topology mistakes in power management solutions.

Coordination of Job Scheduling with Power Management. Our work considers the priority of jobs in power management decisions. In the future, we believe that it is desirable to more tightly integrate job schedulers, which are highly aware of workload priorities, with power managers. Such integration would allow dynamic priorities of different servers (as the jobs running on each server change over time) to be communicated to the power management algorithm quickly, allowing for proactive (as opposed to reactive) power budgeting. On a related note, since existing mechanisms cap power per server, this either requires (1) the scheduler to co-locate jobs of similar priority on a physical server; or (2) researchers to develop a new mechanism that can cap power for individual "virtual partitions" of a server, where each job has its own virtual partition, and where each virtual partition can be assigned its own power budget. While there is research in this space [22], there is no standardized adoption across the industry to leverage these ideas in cloud data centers.

Crossing Provider–User Boundaries for Energy Savings. While public cloud providers strive to save energy to reduce their operating costs, cloud users, particularly for high-performance applications, are often wary of enabling energy-saving mechanisms (such as dynamic frequency scaling) that have potential to impact their application's performance if not managed carefully. In such cases, providers typically turn off the energy-saving features, and pass on the higher cost of operations to users (e.g., higher prices for VM flavors with "guaranteed" performance). Providers need to make the benefits of energy savings visible to users, and preferably share these benefits with users (e.g., by lowering prices), to incentivize the adoption of energy saving mechanisms. Two issues prevent cloud providers from making these benefits visible to users. First, most public cloud environments share a single server with multiple users, but per-user power metering on a server does not currently exist. Second, performance-aware energy-saving solutions that can be adopted in the public cloud (without requiring hooks from the user's workload to the cloud provider's energy management knobs) have received limited attention to date.

8. RELATED WORK

To our knowledge, this work is the first to (1) propose a mechanism to manage power in data centers with multiple power feeds; (2) design a *global priority-aware* power capping system that enables high-priority servers to borrow power from a low-priority server *anywhere* in the data center; and (3) reallocate the stranded power that exists in redundant power infrastructure to servers that need it.

Server & Data Center Power Capping. Power capping first appears in server products in 2006 [1]. Around the same time, Fan et al. [2] observe that data centers rarely consumed their maximum peak power, and could allow up to 39% more servers in the same power infrastructure without throttling. They recommend using power capping as a safety valve, using some amount of throttling to allow for the deployment of an even greater number of servers. Several later works on data center power capping [3, 4, 5, 10, 11, 14, 20, 32, 42-45] propose to effectively increase the server capacity of data centers. For example, Wang et al. [4] propose a hierarchical power capping mechanism to protect power infrastructure while adding more servers to data centers. Facebook's Dynamo [5] extends this work by considering more practical aspects of data center infrastructure, such as circuit breaker characteristics and workload-aware capping actions. All of these works rely on server power capping [6-8] as the underlying mechanism to control power consumption in a data center. However, these server power capping mechanisms control only the sum of power consumption across *all* power supplies of a server, and cannot enforce separate power caps on *individual* power supplies. Therefore, they are inadequate to protect upstream power feeds in redundant power topologies (see Section 3.1). As a result, prior data center power capping methods [3, 4, 5, 10, 11, 14, 20, 32, 42-45], which rely solely on traditional server power capping mechanisms and do not have the context of the redundant power topology, cannot safely control highly-available data centers with *multiple* power feeds, which is one of the important challenges we tackle in this paper. Unlike all of these prior works, our mechanism also effectively utilizes the power that is stranded due to imbalanced load between different power feeds in redundant power infrastructure (see Section 6.3). This is different from the stranded power utilized by prior work [9], which is caused by imbalanced load between different CBs in a single-feed power infrastructure, and which is harnessed in a different way.

Priority-Aware Power Capping. Prior works include some notion of prioritizing the power budgets in capping controllers [1, 4, 5]. However, such priorities are local to a single controller. For example, in Dynamo, the workloads are known in advance and have assigned priorities, and the priority-aware mechanism works only at the leaf controller level, which covers at most "a few hundred" servers [5]. Our proposed solution provides the ability for multiple levels of the capping hierarchy to capture the priority of all child nodes, enabling the consideration of all servers' priorities *across the entire data center* for smarter capping decisions.

Other Mechanisms to Increase Server Capacity and Power Efficiency. Kontorinis et al. [35] propose using energy storage devices to shave peak power demand and allow an increase in server capacity. Wang et al. [36], Hsu et al. [37], and Wallace et al. [38] propose to efficiently use the power infrastructure capacity by performing power-aware workload scheduling, in a way that boosts the capacity of each server. These methods are orthogonal to ours, and can be combined with our proposal to further increase power efficiency. However, using *only* these methods (i.e., without CapMaestro) may not be cost-effective to increase the server capacity of a data center. Energy storage devices such as those used by Kontorinis et al. [35] come with additional cost and space needs, and may need to be replaced after a certain number of charge/discharge cycles. In addition, an energy-storage-only solution cannot handle power peaks that last longer than a few hours. Power-aware workload scheduling [36, 37, 38] puts additional requirements on workloads, such as requiring them to be short-lived [36], repetitive [38], or bear a power consumption pattern lasting several days [37]. In contrast, CapMaestro is designed for existing power infrastructures and can tolerate arbitrary workload characteristics.

To optimize power efficiency, Bai et al. [47] propose a voltage regulator efficiency-aware power management policy, and Kondguli and Huang [48] propose a power-efficient turbo boosting strategy. These works are complementary to ours.

9. CONCLUSION

We present CapMaestro, a new, distributed power management mechanism that uses server power capping effectively to manage the power across an entire data center. CapMaestro is designed to work for redundant power infrastructures in a global priority-aware manner, and it protects against oversubscription at every level of the power distribution hierarchy, while allowing stranded power in the hierarchy to be reallocated to servers that need it. We evaluate a prototype of CapMaestro on real cloud servers to validate its guarantees, and simulate its performance on a large-scale data center environment. We find that for a typical data center where 30% of randomly-selected servers are high-priority, CapMaestro supports 50% more servers than a data center without power capping, and 20% more servers than a data center that uses a state-of-the-art power capping mechanism modified to support redundant power feeds. We discuss a number of remaining important challenges in power management for public clouds, with the hope of inspiring future work in the area.

ACKNOWLEDGMENTS

We thank all anonymous reviewers for their constructive feedback.

REFERENCES

[1] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, “Ensemble-Level Power Management for Dense Blade Servers”, in *ISCA*, 2006.

[2] X. Fan, W. Weber, and L. Barroso, “Power Provisioning for A Warehouse-Sized Computer”, in *ISCA*, 2007.

[3] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, “No "Power" Struggles: Coordinated Multi-Level Power Management for the Data Center”, in *ASPLOS*, 2008.

[4] X. Wang, M. Chen, C. Lefurgy, and T. Keller, “SHIP: A Scalable Hierarchical Power Control Architecture for Large-Scale Data Centers”, in *TPDS*, 2012.

[5] 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 *ISCA*, 2016.

[6] C. Lefurgy, X. Wang and M. Allen-Ware, “Power Capping: A Prelude to Power Shifting”, in *Cluster Computing*, 2008.

[7] Intel Corp., “Intel® Intelligent Power Node Manager 3.0 External Interface Specification Using IPMI”, Document Number 332200-001US, March 2015.

[8] Intel Corp., “Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3B: System Programming Guide, Part 2”, Document Number 253669-060US, Sept. 2016.

[9] L. Ganesh, J. Liu, S. Nath, and F. Zhao, “Unleash Stranded Power in Data Centers with RackPacker”, in *WEED*, 2009.

[10] A. Bhattacharya, D. Culler, A. Kansal, S. Govindan, S. Sankar, “The Need for Speed and Stability in Data Center Power Capping”, in *IGCC*, 2012.

[11] X. Fu, X. Wang, and C. Lefurgy, “How Much Power Oversubscription is Safe and Allowed in Data Centers?”, in *ICAC*, 2011.

[12] M. G. Khatib, Z. Bandic, “PCAP: Performance-Aware Power Capping for the Disk Drive in the Cloud”, in *FAST*, 2016.

[13] M. H. Santriaji, H. Hoffmann, “GRAPE: Minimizing Energy for GPU Applications with Performance Requirements”, in *MICRO*, 2016.

[14] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, “Statistical Profiling-Based Techniques for Effective Power Provisioning in Data Centers”, in *EuroSys*, 2009.

[15] K.J. Åström, and T. Hägglund, *Advanced PID Control*, The Instrumentation, Systems and Automation Society, 2006.

[16] M. Govindan, C. Lefurgy, and A. Dholakia, “Using On-Line Power Modeling for Server Power Capping”, in *WEED*, 2009.

[17] UL LLC, “UL 489: Molded-Case Circuit Breakers, Molded-Case Switches, and Circuit Breaker Enclosures”, 13th edition, Oct. 2016.

[18] Apache Software Foundation, “Apache HTTP Server Project”, https://httpd.apache.org/

[19] Apache Software Foundation, “ab - Apache HTTP Server Benchmarking Tool”, https://httpd.apache.org/docs/2.4/programs/ab.html

[20] R. Azimi, M. Badiei, X. Zhan, N. Li, and S. Reda, “Fast Decentralized Power Capping for Server Clusters”, in *HPCA*, 2017.

[21] National Fire Protection Assn., “NFPA 70: National Electrical Code”, 2017.

[22] H. Lim, A. Kansal, and J. Liu, “Power Budgeting for Virtualized Data Centers”, in *USENIX ATC*, 2011.

[23] D. Wang, S. Govindan, A. Sivasubramaniam, A. Kansal, J. Liu, and B. Khessib, “Underprovisioning Backup Power Infrastructure for Datacenters”, in *ASPLOS*, 2014.

[24] O. Sarood, A. Langer, A. Gupta, and L. Kale, “Maximizing Throughput of Overprovisioned HPC Data Centers Under a Strict Power Budget”, in *SC*, 2014.

[25] H. Zhang and H. Hoffmann, “Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques”, in *ASPLOS*, 2016.

[26] Intel Corp., HP Inc., NEC Corp., Dell Inc., “Intelligent Platform Management Interface (IPMI) Specification v2.0 Rev. 1.1”, 2013.

[27] L. A. Barroso, J. Clidaras, and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, 2nd edition, Morgan Claypool, 2013.

[28] M. Alian, A. H. M. O. Abulila, L. Jindal, D. Kim and N. S. Kim, “NCAP: Network-Driven, Packet Context-Aware Power Management for Client-Server Architecture”, in *HPCA*, 2017.

[29] H. Yang, Q. Chen, M. Riaz, Z. Luan, L. Tang, and J. Mars, “PowerChief: Intelligent Power Allocation for Multi-Stage Applications to Improve Responsiveness on Power Constrained CMP”, in *ISCA*, 2017.

[30] Y. Li, D. Wang, S. Ghose, J. Liu, S. Govindan, S. James, E. Peterson, J. Siegler, R. Ausavarungnirun, and O. Mutlu, “SizeCap: Efficiently Handling Power Surges in Fuel Cell Powered Data Centers”, in *HPCA*, 2016.

[31] C. Li, R. Zhou, and T. Li, “Enabling Distributed Generation Powered Sustainable High-Performance Data Center”, in *HPCA*, 2013.

[32] Z. Tang, H. Zhou, Y. Zhu, R. Tian and J. Yao, “Quantitative Availability Analysis of Hierarchical Datacenter Under Power Oversubscription”, in *SMARTCOMP*, 2017.

[33] Super Micro Computer, Inc., “Embedded BMC/IPMI User’s Guide Revision 2.0”, 2012.

[34] M. Muccini and W. Cook, “Power Consumption Reduction: Hot Spare”, Dell Inc., white paper, 2012.

[35] V. Kontorinis, L. E. Zhang, B. Aksanli, J. Sampson, H. Homayoun, E. Pettis, D. Tullsen, T. S. Rosing, “Managing Distributed UPS Energy for Effective Power Capping in Data Centers”, in *ISCA*, 2012.

[36] G. Wang, S. Wang, B. Luo, W. Shi, Y. Zhu, W. Yang, D. Hu, L. Huang, X. Jin, W. Xu, “Increasing Large-Scale Data Center Capacity by Statistical Power Control”, in *EuroSys*, 2016.

[37] C. Hsu, Q. Deng, J. Mars, L. Tang, “SmoothOperator: Reducing Power Fragmentation and Improving Power Utilization in Large-Scale Datacenters”, in *ASPLOS*, 2018.

[38] S. Wallace, X. Yang, V. Vishwanath, W. E. Allcock, S. Coghlan, M. E. Papka, Z. Lan, “A Data Driven Scheduling Approach for Power Management on HPC Systems”, in *SC*, 2016.

[39] P. E. Bailey, A. Marathe, D. K. Lowenthal, B. Rountree, and M. Schulz, “Finding the Limits of Power-Constrained Application Performance”, in *SC*, 2015.

[40] D. A. Ellsworth, A. D. Malony, B. Rountree, and M. Schulz, “Dynamic Power Sharing for Higher Job Throughput”, in *SC*, 2015.

[41] Y. Li, C. R. Lefurgy, K. Rajamani, M. S. Allen-Ware, G. J. Silva, D. D. Heimsoth, S. Ghose, and O. Mutlu. “CapMaestro: Exploiting Power Redundancy, Data Center-Wide Priorities, and Stranded Power for Boosting Data Center Performance”, IBM Research Report RC25680, Mar. 2018. https://domino.research.ibm.com/library/cyberdig.nsf/1e4115aea78b6e7c85256b360066f0d4/25c5c8ba95611f29852582eb0058ca52

[42] R. Sakamoto, T. Cao, M. Kondo, K. Inoue, M. Ueda, T. Patki, D. Ellsworth, B. Rountree, and M. Schulz, “Production Hardware Overprovisioning: Real-World Performance Optimization Using an Extensible Power-Aware Resource Management Framework”, in *IPDPS*, 2017.

[43] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, “Exploring Hardware Overprovisioning in Power-Constrained, High Performance Computing”, in *ICS*, 2013.

[44] T. Patki, D. K. Lowenthal, B. L. Rountree, M. Schulz, and B. R. de Supinski, “Economic Viability of Hardware Overprovisioning in Power-Constrainted High Performance Computing”, in *E2SC*, 2016.

[45] Z. Zhang, M. Lang, S. Pakin, and S. Fu, “Trapped Capacity: Scheduling under A Power Cap to Maximize Machine-Room Throughput”, in *E2SC*, 2014.

[46] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, “RAPL: Memory Power Estimation and Capping”, in *ISLPED*, 2010.

[47] Y. Bai, V. W. Lee, and E. Ipek, “Voltage Regulator Efficiency Aware Power Management”, in *ASPLOS*, 2017.

[48] S. Kondguli and M. Huang, “A Case for a More Effective, Power-Efficient Turbo Boosting”, in *TACO*, 2018.