2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture

Dynamo: Facebook's Data Center-Wide Power Management System

Qiang Wu, Qingyuan Deng, Lakshmi Ganesh, Chang-Hong Hsu*,
Yun Jin, Sanjeev Kumar†, Bin Li, Justin Meza, and Yee Jiun Song
Facebook, Inc. **University of Michigan*

***Abstract*—Data center power is a scarce resource that often goes underutilized due to conservative planning. This is because the penalty for overloading the data center power delivery hierarchy and tripping a circuit breaker is very high, potentially causing long service outages. Recently, dynamic server power capping, which limits the amount of power consumed by a server, has been proposed and studied as a way to reduce this penalty, enabling more aggressive utilization of provisioned data center power. However, no real at-scale solution for data center-wide power monitoring and control has been presented in the literature.**

In this paper, we describe Dynamo – a data center-wide power management system that monitors the entire power hierarchy and makes coordinated control decisions to safely and efficiently use provisioned data center power. Dynamo has been developed and deployed across all of Facebook's data centers for the past three years. Our key insight is that in real-world data centers, different power and performance constraints at different levels in the power hierarchy necessitate coordinated data center-wide power management.

We make three main contributions. First, to understand the design space of Dynamo, we provide a characterization of power variation in data centers running a diverse set of modern workloads. This characterization uses fine-grained power samples from tens of thousands of servers and spanning a period of over six months. Second, we present the detailed design of Dynamo. Our design addresses several key issues not addressed by previous simulation-based studies. Third, the proposed techniques and design have been deployed and evaluated in large scale data centers serving billions of users. We present production results showing that Dynamo has prevented 18 potential power outages in the past 6 months due to unexpected power surges; that Dynamo enables optimizations leading to a 13% performance boost for a production Hadoop cluster and a nearly 40% performance increase for a search cluster; and that Dynamo has already enabled an 8% increase in the power capacity utilization of one of our data centers with more aggressive power subscription measures underway.

***Keywords*—data center; power; management.**

I. Introduction

Warehouse-scale data centers consist of many thousands of machines running a diverse set of workloads and comprise the foundation of the modern web. The power delivery infrastructure supplying these data centers is equipped with *power breakers* designed to protect the data center from damage due to electrical surges. While tripping a power breaker ultimately protects the physical infrastructure of a data center, its application-level effects can be disastrous, leading to long service outages at worst and degraded user experience at best.

Given how severe the outcomes of tripping a power breaker are, data center operators have traditionally taken a conservative approach by *over-provisioning* data center power, provisioning for worst-case power consumption, and further adding large power buffers [1], [2]. While such an approach ensures safety and reliability with high confidence, it is wasteful in terms of power infrastructure utilization – a scarce data center resource. For example, it may take several years to construct a new power delivery infrastructure and every megawatt of power capacity can cost around 10 to 20 million USD [2], [3].

Under-utilizing data center power is especially inefficient because power is frequently the bottleneck resource limiting the number of servers that a data center can house. It is even more so with the recent trend of increasing server power density [4], [5]. Figure 1 shows that server peak power consumption nearly doubled going from the 2011 server (24-core Westmere-based) to the 2015 server (48-core Haswell-based) at Facebook. This trend has led to the proliferation of *ghost spaces* in data centers: unused, and unusable, space [6].

To help improve data center efficiency, *over-subscription* of data center power has been proposed in recent years [1], [6], [7]. With over-subscription, the planned peak data center power demand is *intentionally* allowed to surpass data center power supply, under the assumption that correlated spikes in server power consumption are infrequent. However, this exposes data centers to the risk of tripping power breakers due to highly unpredictable power spikes (e.g., a natural disaster or a special event that causes a surge in user activity for a service). To make matters worse, a power failure in one data center could cause a redistribution of load to other data centers, tripping their power breakers and leading to a cascading power failure event.

Therefore, in order to achieve both power safety and

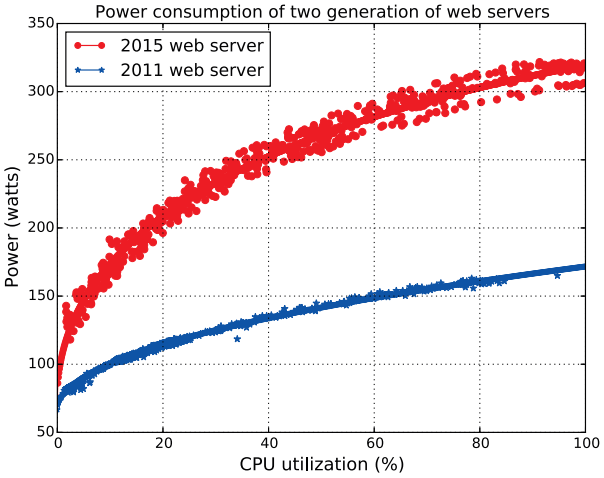

Figure 1. The measured power consumption (in watts) as a function of server CPU utilization for two generations of web servers used at Facebook. The ⋆ data points were measured from a 24-core Westmere-based web server (24×L5639@2.13GHz, 12GB RAM, 2×1G NIC) from 2011, while the • data points were measured from a 48-core Haswell-based web server (48×E5-2678v3@2.50GHz, 32GB RAM, 1×10G NIC) from 2015. Both servers were running a real web server workload. We varied the server processor utilization by changing the rate of requests sent to the server. Note that the 2015 server power was measured using an on-board power sensor while the 2011 server power was measured using a Yokogawa power meter.

* Work was performed while employed by Facebook, Inc.
† Currently with Uber, Inc.

1063-6897/16 $31.00 © 2016 IEEE
DOI 10.1109/ISCA.2016.48

improved power utilization with over-subscription, *power capping*, or peak power management techniques, have been proposed [4], [8]. These techniques (1) continually monitor server power consumption and (2) use processor and memory dynamic voltage and frequency scaling (DVFS) to suppress server power consumption if it approaches its power or thermal limit.

Most prior work focused on server-level [8], [9], [10] or ensemble-level [4] power management. They typically used hardware P-states [11] or DVFS to control peak power or thermal characteristics for individual or small groups of servers, with control actions decided locally in isolation. There have been fewer studies on data center-wide power management to monitor all levels of the power hierarchy and make coordinated control decisions. Some recent work [12], [13] looked into the problem of data center-wide power management and the issues of coordination across different levels of controllers. However, their proposed techniques and design were limited because of the simplified system setup in their simulation-based studies, which were based on either pure simulation or a small test-bed of fewer than 10 servers. These prior works did not address many key issues for data center-wide power management in a real production environment with tens or hundreds of thousands of servers.

In this paper, we describe Dynamo – a data center-wide power management system that monitors the entire power hierarchy and makes coordinated control decisions to safely and efficiently use provisioned data center power. Our key insight is that in real-world data centers, different power and performance constraints at different levels in the power hierarchy necessitate coordinated, data center-wide power management. We make three main contributions:

1. To understand the design space of Dynamo, we provide a characterization of power variation in data centers running a diverse set of modern workloads. Using fine-grained power samples from tens of thousands servers for over six months, we quantify the power variation patterns across different levels of aggregation (from Rack to MSB) and across different time scales (from a few seconds to tens of minutes). Based on these results, as well as our study of power breaker characteristics, we find that to prevent real-world power failures, the controller power reading cycle should be fast – on the order of a few seconds – as opposed to minutes as suggested by previous work.

2. We describe the design of a data center-wide power management system in a real production environment. Our design addresses several key issues not dealt with by previous simulation-based studies, such as (1) scalable communication between controller and controllee, (2) application- and service-aware capping actions, and (3) coordination of multiple controller instances with heterogeneous workload and data dependence.

3. We deploy and evaluate Dynamo in large scale data centers serving billions of users. We report a rich set of results from real-life power capping events during critical power limiting scenarios. We also describe real use cases where Dynamo enables optimizations (such as Turbo Boost and aggressive power over-subscription) leading to performance and capacity improvements. For example, Dynamo can improve the performance of a production Hadoop cluster by up to 13% and has enabled us to accommodate 8% more servers under the same power constraints via more aggressive power budgeting in an existing data center.

Dynamo has been deployed across all of Facebook's data centers for the past three years. In the rest of this paper, we describe its design and share how it has enabled us to greatly improve our data center power utilization.

II. Background

In this section we describe what data center power delivery infrastructure is, what it means to *oversubscribe* its power, and under what conditions such oversubscription can cause power outages. We also discuss the implications these factors have on the design of a data center-wide power management system.

A. Data Center Power Delivery

Figure 2 shows the power delivery hierarchy in a typical Facebook data center, based on Open Compute Project (OCP) specifications [14]. The local power utility supplies the data center with 30 MW of power. An on-site power sub-station feeds the utility power to the *Main Switch Boards (MSBs)*. Each MSB is rated at 2.5 MW for IT equipment power and has a standby generator that provides power in the event of a utility outage.

A data center typically spans four rooms, called *suites*, where racks of servers are arranged in rows. Up to four MSBs provide power to each suite. In turn, each MSB supplies up to four 1.25 MW *Switch Boards (SBs)*. From each SB, power is fed to the 190 KW *Reactive Power Panels (RPPs)* stationed at the end of each row of racks.

Each RPP supplies power to (1) the racks in its row and (2) a set of *Direct Current Uninterruptible Power Supplies (DCUPS)*. Each DCUPS provides 90 s of power backup to six racks. The rack power shelf is rated at 12.6 KW. Depending on the server specifications, there can be anywhere between 9 and 42 servers per rack[1].

[1]Here we describe the typical Facebook-owned data center based on OCP specifications [14]; Facebook also leases data centers where the power delivery hierarchy matches more traditional ones described in literature [1]. The traditional model uses *Power Distribution Units (PDUs)* and *PDU Breakers* in place of SBs and RPPs.

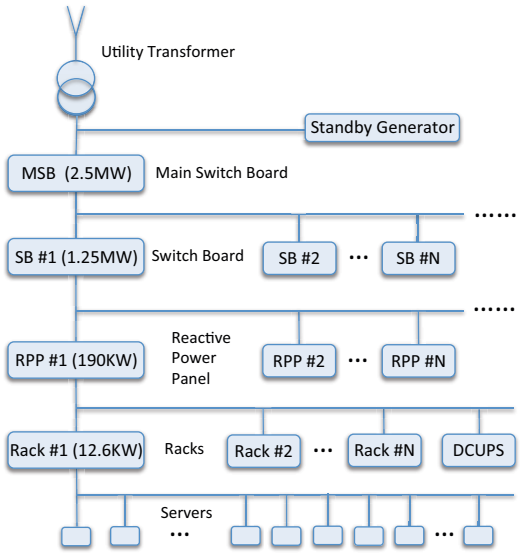

Figure 2. Typical Facebook data center power delivery infrastructure [14].

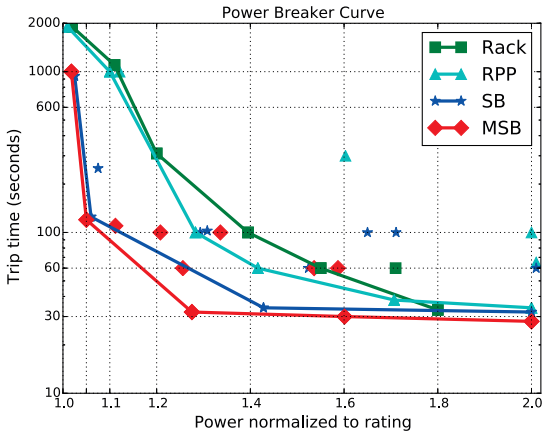

Figure 3. Power breaker trip time as a function of power usage. The dots represent raw data based on manufacturer testing for different power breakers used at Facebook. The lines show the lower-bound on trip time per device type.

Notice that power is *oversubscribed* at each level: a power device supplies less power than its children draw at peak. For example, an MSB supplies 2.5 MW to up to four SBs that draw $4 \times 1.25 = 5$ MW at peak. Each power device is equipped with a circuit breaker that *trips* if the device's power draw exceeds the breaker's rated power. But breakers do not trip instantly. We measured the amount of time it took to trip the breakers at each level of our power delivery hierarchy under different amounts of power overdraw, shown in Figure 3 (note the figure's logarithmic y-axis). Our results corroborate measurements reported in prior literature [7].

We find that a breaker trips when both (1) the current through the breaker exceeds the breaker's rated current and (2) the breaker sustains the overdrawn power for a period of time *inversely proportional to the overdraw amount*. In other words, *though circuit breakers trip quickly under large power spikes, they sustain low amounts of overdrawn power for long periods of time*. For example, RPPs and Racks sustain a 10% power overdraw for around 17 minutes. In addition, as Figure 3 shows, *lower-level devices in the power delivery hierarchy sustain relatively more power overdraw than higher-level devices*. For example an RPP sustains a 40% power overdraw for around 60 s while an MSB sustains only a 15% power overdraw for the same period of time.

Because power oversubscription occurs at every level of the power delivery hierarchy, *it is insufficient for power capping techniques to monitor any single device or subset of devices in a data center*. Instead, techniques must take a holistic approach, coordinating action across all devices in the power delivery hierarchy. In addition, because some slack exists in how long it takes circuit breakers to trip, *opportunities exist for minimizing the impact of server performance while still ensuring power safety*. These observations inform the design of Dynamo that we discuss in Section III.

B. Power Variation Characterization

In Section II-A, we observed that power breaker trip time varies as a function of power overdraw. A key design consideration for power capping techniques, then, is how fast to respond to power overdraw in order to guarantee protection from tripping circuit breakers. To do this, we must answer the question, *how quickly does power draw change in a real-world production data center?*

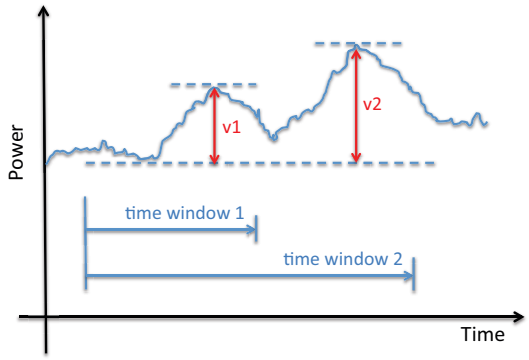

Figure 4. An illustration of calculated power variation for a time window. The maximum power variation is the difference between the maximum and minimum power values in the time window. Here, v1 and v2 are the maximum power variations for time windows 1 and 2, respectively.

Since we can not safely overload the power delivery hierarchy in our data centers, we instead measure and extrapolate power variation to power-oversubscribed scenarios. We next present a large-scale study characterizing the power variations of servers at Facebook. We collected fine-grained power values (every 3 seconds) for every server in one data center suite (roughly 30 K servers) for over six months. We also examine coarser-grained power values (every 1 minute) for all servers in all our data centers (on the order of hundreds of thousands) for nearly 3 years.

To quantify power variation, we define the *power slope*, which measures the rate at which power can increase in a specific time window (from 3 seconds to 600 seconds) for different levels of the power hierarchy (Figure 2). Figure 4 illustrates how the metrics are calculated. For each time window, we calculate the worst-case power variation as the difference of the maximum and minimum power values in that time window.

We analyze the fine-grained power data for each level of the power hierarchy (rack, RPP, SB, and MSB) over six months. To simplify the analysis, we partition the data into pieces, study data from two-week time periods, and combine results from multiple time periods. Figure 5 shows the summarized results on power variations and allows us to make the following two observations. First, larger time windows have generally larger power variations. Second, the higher the power hierarchy level, the smaller the relative power variation, due to load multiplexing. For example, at the rack level, the worst-case power variations range from 10% to 50% for different time windows, while at the MSB level, the worst-case power variations range from 1% to 6% for the same time window.

A third observation is that power variations also depend on the application. We randomly selected a group of servers from several services at Facebook (web server, cache, MySQL database, news feed, hadoop, and f4/photo storage [15]; with 30 servers per service) and conducted a similar analysis (for one specific time window of 60 s). Figure 6 summarizes the results. We see that different applications have different power variation characteristics. For example, servers running f4/photo storage have the lowest median (50^{th} percentile,

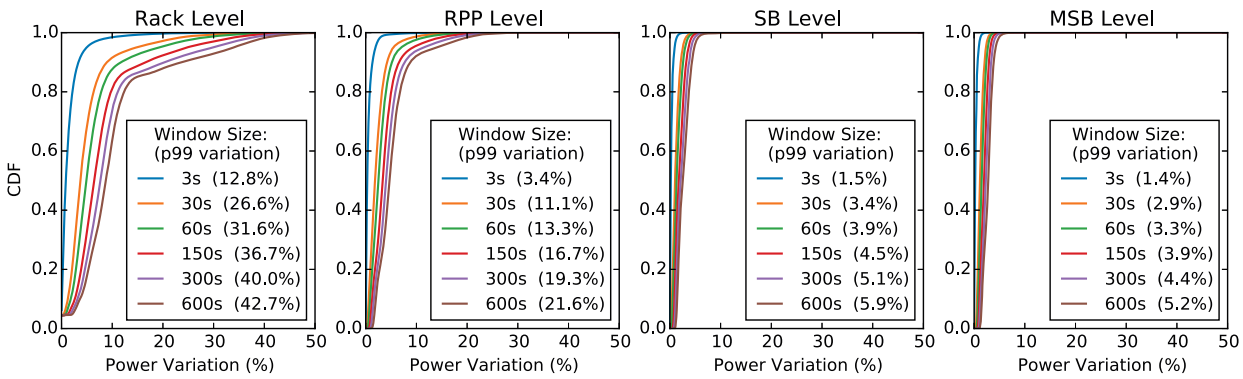

Figure 5. The measured power variations over different time windows for power devices at the racks, RPPs, SBs, and MSBs. The x-axis is the power variation normalized to the average power during peak hours. The y-axis is the cumulative distribution function. Lines of different colors represent different time windows (from 3 s to 600 s). For convenience, we also list the 99$^{\text{th}}$ percentile (p99) power variation values for each case.

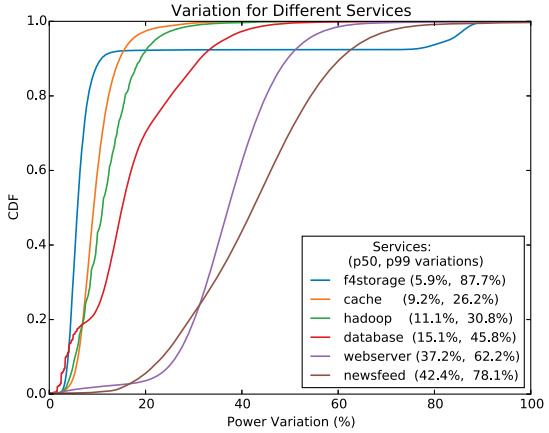

Figure 6. The measured power variations for several services in Facebook at the server level. The default time window is 60 s. For convenience, we also show the median (50$^{\text{th}}$ percentile, or p50) and p99 power variation values for each case.

or p50) variations but the highest p99 variations among all services we have studied.

C. Design Implications

Combining the observations from Sections II-A and II-B, we derive the following design implications for a data center-wide power management system.

Sub-minute power sampling. Such a system needs to *sample* power at a sub-minute interval. Figure 5 shows that 3% (at MSB level) to 30% (at rack level) power demand increases have been observed within a 60 s interval. Such swings in power demand are large enough to trip a breaker within a few minutes according to Figure 3.

2-minute power capping time. Such a system also needs to *react* to power demand spikes in no more than 2 minutes (and potentially much less to guarantee safety). Figure 3 shows that breakers trip for a ∼5% MSB power overdraw in as little as 2-minutes. Within this 2-minute interval a power management system must issue appropriate capping commands and ensure that power settles.

We next discuss Dynamo, our data center-wide power management system. We design Dynamo to sample data at the granularity of a few seconds and conservatively target 10 s of time for control actions and power settling time. This is an important distinction from prior work that sampled power at a coarser granularity of several minutes [1].

III. Dynamo Architecture

This section describes the design of Dynamo. We start with an overview and then describe in detail the components that Dynamo comprises: the *agent*, the *leaf power controller*, and the *higher-level power controllers*.

A. Overview of Dynamo's Design

As mentioned in Section I, our goal is to design a data center-wide power management system that monitors the entire power hierarchy. For a practical system to be deployed to tens or hundreds of thousands of servers, it needs to be (1) efficient and reliable and (2) scalable to handle a power hierarchy of extremely large size.

At a high level, Dynamo has two major components:

Dynamo agent. The agent is a light-weight program deployed to every server in the data center. It reads power, executes power capping/uncapping commands, and communicates with the controllers (discussed next). Since a Dynamo agent needs to run on every server, it is designed to be as simple as possible. We place most of the intelligence of the system in the controller. As a result, Dynamo agents do not communicate with one another and only communicate with Dynamo controllers.

Dynamo controllers. The controllers run on a group of dedicated servers, monitor data from the Dynamo agents under their control, and are responsible for protecting data center power devices. We take a distributed approach in the design of the Dynamo controllers. For every physical power device in the hierarchy that needs protection there is a matching controller instance monitoring and controlling (directly or indirectly) the set of downstream servers for that device. In this way, there is a hierarchy of Dynamo controllers that mirrors the topology of the data center's power hierarchy. Multiple levels of controllers coordinate to ensure the safety of the entire power hierarchy.

Figure 7 illustrates how Dynamo's major components interact with each other. Each power device at the lowest level is assigned a *leaf power controller*. The leaf power controller communicates directly with Dynamo agents on all downstream servers of that power device. We use the Thrift [16] remote procedure call (RPC) service for efficient and reliable

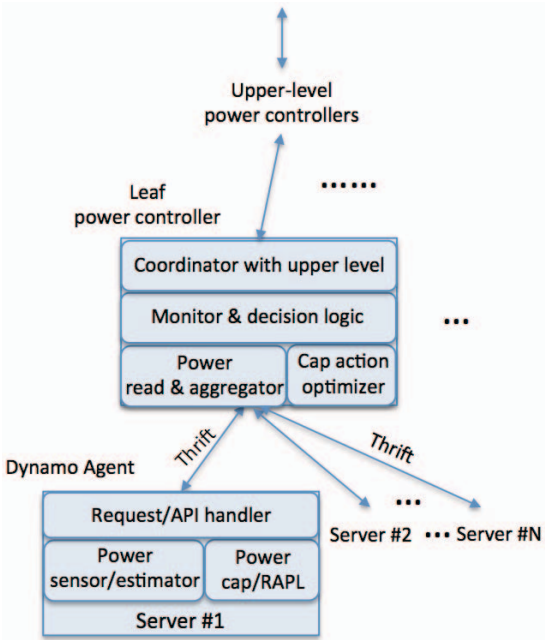

Figure 7. An illustration of how Dynamo's major components interact with each other. Dynamo has two major components: the agent, which runs on every server at Facebook; and the controller, which monitors the power of each device in the power delivery hierarchy.

communication between controllers and agents. At each *sampling interval* a leaf controller reads and aggregates power for all downstream servers and compares the aggregated power to the device's power limit. The decision logic (which we describe in Section III-C2) then decides whether capping or uncapping is needed. In the case of capping, the leaf controller will also try to optimize the capping actions to minimize the overall performance impact caused by power capping.

Similarly, each power device at a non-leaf level is assigned an *upper-level power controller*. The upper-level controller indirectly monitors and controls its downstream servers by communicating with leaf controllers or other downstream upper-level controllers. Controller communication ensures that controller decisions converge and upper-level power devices are protected, and can be through Thrift if the controller instances are fully distributed or through shared memory if the controllers are running on the same server.

Next, we will describe each of the major components of Dynamo in detail.

B. Agent Design

Dynamo agent is a light-weight program running on every server in a data center. Figure 8 shows the overall structure and work flow for Dynamo agent. At a high level, Dynamo agent functions like a request handler daemon. It continuously listens for requests sent remotely from the leaf controller. We choose the Thrift RPC service as the main communication protocol for two reasons: first, it is an efficient RPC mechanism that allows decisions to propagate quickly from the leaf controller to the agents; and second, it is highly scalable and has been proven to handle communication among many thousands servers [16].

There are two basic types of requests a Dynamo agent handles:

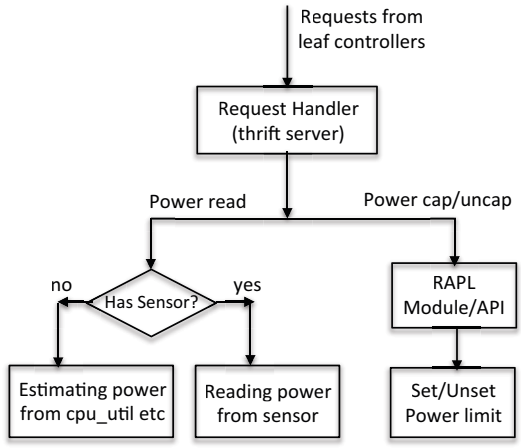

Figure 8. A block diagram for Dynamo agent showing the overall structure and workflow.

Power read. The agent reads and returns the current power consumption of the host server. If possible, it also returns the breakdown of the power (such as fan power, socket power, AC-DC power loss, etc.). Nearly all new servers (2011 or newer) at Facebook are equipped with an on-board power sensor, which provides accurate power readings. Internally, the agent talks to the sensor firmware to get the power reading and breakdowns. For a small group of servers without power sensors, we build a power estimation model similar to [17] by measuring server power with respect to CPU utilization with a Yokogawa power meter [18]. Once a server's power model is built, the agent estimates its power on-the-fly using system statistics such as CPU utilization, memory traffic, and network traffic.

Power capping/uncapping. The agent sets or unsets the power limit for the host server based on capping or uncapping requests sent from the leaf controller. It also returns the status of the operation to the leaf controller to let it know whether the operation was executed successfully. Internally, agents communicate with the Intel running average power limiting (RAPL) module to implement setting or unsetting the power limit. RAPL is a single-server level power control mechanism to enforce total system power budgeting [19], [20]. Communicating with RAPL is platform-specific – we either update a machine status register (MSR) directly or, when available, call the API provided by the on-board node manager through IPMI [19], [21].

To evaluate the effectiveness of the agent's operation, we examined how long it takes for the agent to cap or uncap power to a target level for a single server. As shown in Figure 9, we found that once a RAPL capping/uncapping command is issued, it takes about two seconds for it to take effect on the target server and stabilize. This result implies that the controller, which reads power from the agent, must sample at larger than a two-second granularity in order to get stable power readings and aggregations.

C. Leaf Power Controller Design

In this section, we discuss the most basic controller type – the leaf power controller, which communicates with the agents under the corresponding power device. Its functionality consists of three parts: (1) power reading and aggregation, (2) monitoring and making capping decisions, and (3)

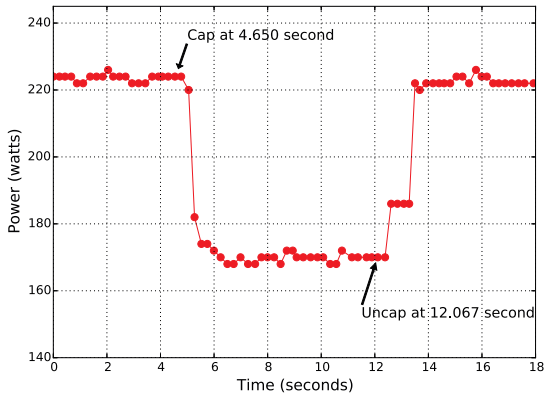

Figure 9. The measured single-server power capping and uncapping test results using Dynamo agent and RAPL.

choosing the optimal set of servers to cap in order to minimize performance impact. We discuss each of these in turn next.

1) Power Reading and Aggregation: The leaf power controller talks to agents to get periodic power consumption data for all servers under the power device. It aggregates the power readings to calculate the total power consumption for the servers connected to the power device.

Specifically, a leaf power controller periodically broadcasts power pull requests over Thrift to all servers under its purview. Recall from Section II-B that we require a sub-minute power sampling interval (and faster sampling leads to potentially better control actions). On the other hand, we saw in Section III-B that the sampling period needs to be greater than 2 seconds in order to get stable power readings. We therefore pick 3 seconds as the pulling cycle for the leaf power controller to get both stable readings and fast reaction times.

Some power breakers provide power readings directly, so why aggregate power readings from servers instead of just using the breaker readings? There are two main reasons for our design choice. First, even when a breaker provides power readings, the sampling frequency is usually not fast enough for our use case. For example, for power breakers used at Facebook, the available sampling frequency is on the order of minutes and there is no easy way to get finer-grained power samples. Second, we need to know individual server power consumption in order to figure out the best power capping decisions for each server and send out power capping commands to the right servers (we describe specifically our policy in Section III-C3). Therefore, for Dynamo, we use the power breaker readings only for validating that the aggregated power from servers is correct.

A leaf power controller typically pulls power from a few hundred servers or more. It may happen that power pulling fails or times out for some servers. To handle this, the leaf power controller uses additional information (server type, server status, application type, etc.) about the group of servers and attempts to estimate the power reading for the failed servers using power readings from neighboring servers running similar workloads. Further, in the rare case that more than 20% of the servers in the group have power reading failures, the leaf power controller will treat the power aggregation as invalid and send an alert for human intervention. Finally, for non-server components (such as the top-of-rack switch) drawing power from the same breaker, the leaf power controller will try to pull power directly from the component if available, and use an estimate if not.

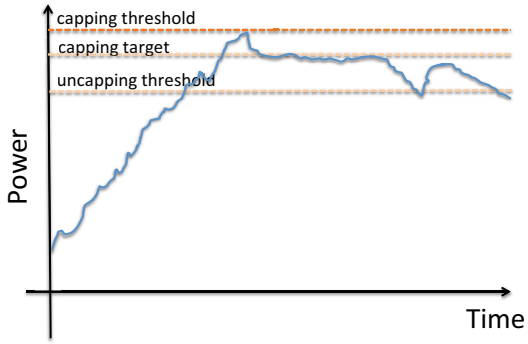

Figure 10. An illustration of the three-band power capping and uncapping algorithm.

2) Power Capping Decisions: Once the leaf power controller aggregates power readings from servers to calculate the total power consumption for a power device, it compares it to the limit of the power breaker (see Section III-D for details on computing the power breaker limit).

The leaf power controller uses a *three-band* algorithm to make capping or uncapping decisions, shown in Figure 10. There is a *capping threshold* (the top band), which is typically 99% of the limit of the breaker. When the aggregated power exceeds the capping threshold, power capping logic attempts to bring down the power consumption to the level of the *capping target* (the middle band). The capping target is conservatively chosen to be 5% below the breaker limit for safety (similar to the conservative single-step actuation in [22]). To avoid oscillations, there is also a lower *uncapping threshold* (the bottom band) such that power uncapping will be triggered only if the aggregated power falls below it.

In practice, the three-band algorithm efficiently eliminates control oscillations while making the capping response time fast enough to handle sudden power surges in our data centers. The algorithm is also flexible – we can configure the capping and uncapping thresholds on a per-controller basis enabling customizable trade-offs between power-efficiency and performance at different levels of the power delivery hierarchy.

3) Performance-Aware Power Capping: Though we prioritize safety and treat power capping as an emergency action, we still want to reduce its effect on application performance as much as possible. Therefore, a key function of the leaf power controller is to choose the right set of servers to cap (and their capping targets) to reduce performance impact. The leaf power controller uses meta-data about all the servers it controls, in conjunction with their power consumption history, to choose appropriate capping actions.

We start by categorizing Facebook services into a predefined set of *priority groups*, where higher priority corresponds potentially higher performance impact due to power capping. For example, cache servers [23] belongs to a higher priority group than web servers or news feed servers because a small number of cache servers may affect a large number of users and have more direct impact on end-to-end performance. Each priority group has its own *service-level agreement (SLA)* in terms of the lowest allowable power cap.

When a leaf power controller needs to cap power, it first calculates the *total-power-cut*: the difference between the current aggregated power and the capping target (shown in

Figure 10). Then, it tries to distribute the total-power-cut to its servers from services belonging to the lowest priority group first. If capping the servers in the lowest priority group does not absorb all of the total-power-cut, the leaf power controller picks servers from services belonging to the second-lowest priority group, and so on.

Within servers from services belonging to the same priority group, the leaf power controller uses a *high-bucket-first* algorithm to decide exactly how much of a power cut each server should take. Analogous to tax brackets, it first groups servers into different buckets based on their current power consumption and then distributes the total-power-cut among servers in the highest bucket first. The rationale is to punish first servers consuming more power (likely caused by application regressions, unexpected software behavior consuming system resources, etc.). If the highest bucket is not enough to fulfill the power cut, it will then expand to include servers in the next bucket, and so on, until either the total-power-cut can be satisfied or it hits the SLA lower bound (i.e., the lowest power cap allowed for the current priority group). Within the bucket, all servers will get an even amount of power cut. Based on our experience, we find a bucket size between 10 and 30 W works well for most servers. In our current configuration a bucket size of 20 W is used.

Once the leaf power controller calculates the allocated power-cut for each server (which, in total, adds up to the total-power-cut), it then computes the power cap for each server as its current power value less its power-cut. For example, if a server is currently consuming 250 W and its allocated power-cut is 30 W, its power cap will be set to 220 W. Finally, the leaf power controller sends capping requests, along with these capping targets, to all affected servers.

D. Upper-Level Power Controller Design/Coordination

As mentioned in Section III-A, a hierarchy of Dynamo controllers exists, mirroring the power delivery hierarchy. For every power device that needs to be protected, there is a controller instance responsible for monitoring and protecting the device. In theory, the controllers can be fully distributed with each controller instance being an independent binary and communication between instances occurring via Thrift. However, since most controllers are relatively lightweight, it is also possible to consolidate them to reduce their resource footprint.

We have described the most basic type of Dynamo controller – the leaf power controller. The rest of this section will focus on upper-level power controllers. The upper-level power controller shares the same functions as the leaf controller in term of:

Power reading and aggregation. The upper-level power controller periodically pulls power readings. The difference is that these readings are pulled from downstream controllers, rather than directly from servers. To ensure control stability, the pulling cycle for the upper-level controller is longer than the settling time of the downstream leaf controller [24]. In our implementation, the upper-level power controller uses a pulling cycle of 9 seconds, which is 3× the pulling cycle of the leaf power controller.

Capping and uncapping decisions. Once the upper-level power controller gets the aggregated total power, the capping decision logic uses the same three-band algorithm shown in Figure 10 to decide when to cap or uncap.

An important concern is that, since both the leaf power controller and the upper-level power controller have independent capping/uncapping logic, they must coordinate to make sure their capping actions will not conflict or negatively interact with each other.

We use a *punish-offender-first* algorithm to coordinate and choose the right actions for the upper-level power controller. The basic idea is that, when an upper-level power controller exceeds its power limit and triggers capping, it inspects its children controllers and punishes the offenders first. Here the offender is defined based on the power quota (planned peak power consumption) for a device. To illustrate the punish-offender-first algorithm, let us assume a parent device $P1$ has two children devices $C1$ and $C2$ with

$P1$: physical power limit = 300 KW

$C1$: physical power limit = 200 KW, power quota = 150 KW

$C2$: physical power limit = 200 KW, power quota = 150 KW

Notice that the children may individually consume as much as 200 KW so long as their combined power usage is below 300 KW. However, when this is violated, say

$C1$: power usage = 190 KW

$C2$: power usage = 130 KW,

the upper-level controller for $P1$ will see a power usage of 320 KW exceeding its limit of 300 KW. It will then use the punish-offender-first algorithm to distribute the needed power cut (20 KW for this example) among its children controllers. For this example, $C1$ is the offender as its current power usage is above its power quota. The needed 20 KW power cut will be given to $C1$. To do so, the upper-level controller for $P1$ will send a *contractual power limit* of 170 KW to the controller for $C1$. And we get

$C1$: physical power limit = 200 KW

contractual power limit = 170 KW.

The controller for $C1$ will use the minimum of the physical and contractual limits to make capping decisions. Assuming it is a leaf controller, it will find the optimal set of servers to cap as described in Section III-C3. After the capping action propagates to the downstream servers, we expect $C1$ in the next control cycle to satisfy

$C1$: power usage $\leq$ 170 KW.

In case the controller for $C1$ is an upper-level controller, it will then recursively propagate its decisions to downstream controllers via more contractual power limits.

Note that if $P1$ had to choose among multiple offenders, it would distribute the power cut among all downstream offenders using a high-bucket-first algorithm as described in Section III-C3. Finally, each controller chooses the minimum of the its individual capping decision and that propagated from its parent controller.

E. Dynamo Design Discussion

We conclude this section by discussing several additional issues related to Dynamo's design.

Fault tolerance. Since Dynamo manages the power safety of our data centers, we designed it to be fault tolerant and resilient against agent and controller failures. First, a script periodically checks the health of an agent and restarts the agents in case the agent crashes. This occurs in addition to other generic auto-remediation systems at Facebook that handle hardware failures and job rescheduling. Second, as

discussed in III-C1, a controller can tolerate a certain threshold of power pulling failures (using estimated power based on server meta-data). If the number of power pulling failures exceeds a threshold, a controller will avoid taking false positive actions. It will instead send an alarm for a human operator to intervene. In case a controller crashes, we use a redundant backup controller that resides in a different location and can take control as soon as the primary controller fails.

Networking hardware. Besides servers, there are also network devices in our data centers. Network devices consume relatively small amounts of power in Facebook data centers, usually a low single digit percentage as compared to servers. Currently, Dynamo monitors – but does not control – the power consumption of network devices. This is because the network hardware in our data centers does not yet support RAPL-like power capping. Our current approach is to treat those network devices separately (and budget power for them accordingly). However, in case future network devices support capping, Dynamo can be easily extended to control network devices as well.

Algorithm selection. Our primary objective was to develop a reliable power management system at scale. We made the decision to use a simple three-band algorithm (Section III-C2) to help us quickly iterate on the design process and easily identify issues when testing in production. In the future, we may explore more complex power capping algorithms.

IV. Results: Dynamo In Action

Dynamo was developed at Facebook and gradually deployed to all of its production data centers over the past three years. For the particular setup used at Facebook, we configure RPPs or PDU Breakers (depending on the data center type) as the leaf controllers and skip rack-level power monitoring[2]. As a result, the implementation and deployment of Dynamo has been simplified as the number of leaf controllers has been reduced (from one per rack to one per RPP or PDU Breaker), even though the leaf controller now needs to handle a larger group of servers (between a few hundred to a thousand). To optimize for performance in our implementation, all controller instances for neighboring devices in a data center suite are consolidated into one binary with each controller instance being a thread (there are around 100 threads in total). Different controller binaries run on separate and dedicated Dynamo servers to avoid a single point of failure.

In the rest of this section, we present results to display Dynamo operations in production.

A. Power Capping

Figure 11 shows a power capping event that happened at the leaf controller level in one front-end cluster located in Ashburn, Virginia. The leaf controller was protecting a PDU Breaker (rated at 127.5 KW) powering several hundred front-end web servers. From 8:00 AM to 10:30 AM, the total power of these servers increased steadily as a result of normal daily traffic increase at Facebook. Around 10:40 AM, however, a production load test was started with more user traffic shifting to this cluster. Notice that the load testing drove the power consumption of the PDU Breaker to rapidly approach its capping threshold of 127 KW, and finally exceed it around 11:15 AM. As a result, power capping was triggered. The leaf power controller successfully throttled power to a safe level within about 6 s, and then held it at a level slightly below the capping target of 126 KW until 11:45 AM, when the production load testing finished and the traffic to this cluster started to return to normal. Around 12:00 PM the total power dropped below the leaf power controller's uncapping threshold, which triggered power uncapping.

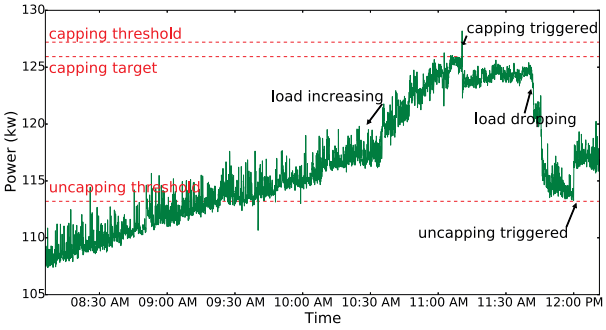

Figure 11. Power measurement showing power capping and uncapping for a row of servers in one front-end cluster located in Ashburn, Virginia.

Figure 12 shows another power capping instance – one that prevented a potentially severe power outage. This instance began with an unplanned site outage, which affected several Facebook data centers. Operational issues during the recovery process resulted in one of the data centers in Altoona, Iowa receiving significantly higher-than-normal traffic, further exacerbated by many servers starting up simultaneously during the recovery. The result was a dramatic power surge in that data center. In particular, the power consumption of one SB rose to $1.3\times$ its normal daily peak, approaching its physical breaker limit. The upper-level power controller guarding this SB kicked in and capped three offender rows to avoid tripping the SB power breaker. (Note that, during a vulnerable time of recovery, another SB-level power outage could have significantly worsened the situation or even caused cascading failures.)

Figure 12 shows the detailed sequence of events. The unplanned outage started around 12:00 PM, and caused a sharp power drop in the SB over the next ten minutes. This was followed by a couple of unsuccessful partial recovery attempts, which caused the power to oscillate over the next 30 minutes or so. Then, around 12:48 PM, the recovery was successful and a large amount of traffic started to flow in, which caused a large power surge. Shortly after 12:48 PM, the SB-level Dynamo controller kicked in, capped three offender rows, and held the power steadily below the limit. This lasted for another 20 minutes until the power dropped to a safe level, whereupon Dynamo started to uncap. We see that after uncapping, the power actually bounced back a bit but was still below the limit. Finally, around 1:35 PM, more traffic was shifted to other data centers and the load to this data center returned to normal around 2:00 PM.

An important question is *how does power capping affect performance or latency?* For the above power capping events on front-end clusters, the observed performance degradation was negligible. This is because both events only affected a small subset of servers in certain rows and request load

[2]It is simply a Facebook infrastructure reality that under the current 30 MW data center design and the building space available, the available rack power is always over-provisioned and the rack is not a power delivery bottleneck. Dynamo will, however, work equally well regardless how rack power is configured.

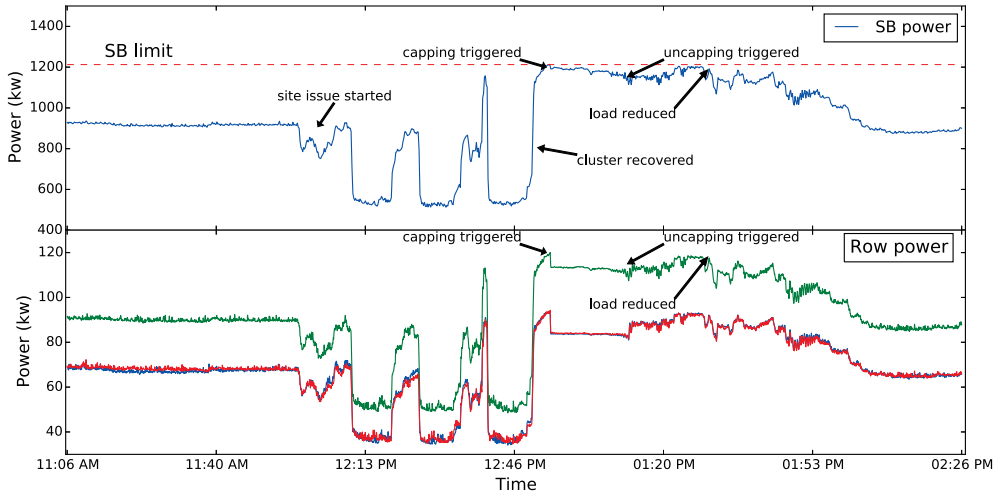

Figure 12. A real-world case study of how Dynamo prevented a potential power outage. A power surge occurred during recovery from an unplanned site issue and led one SB in Facebook's Altoona, Iowa data center to exceed its power limit. An upper-level power controller kicked in around 12:48 PM and three offender rows/RPPs got capped. The upper graph is the power consumption of the SB, while the lower graph is the power consumption for the three rows/RPPs.

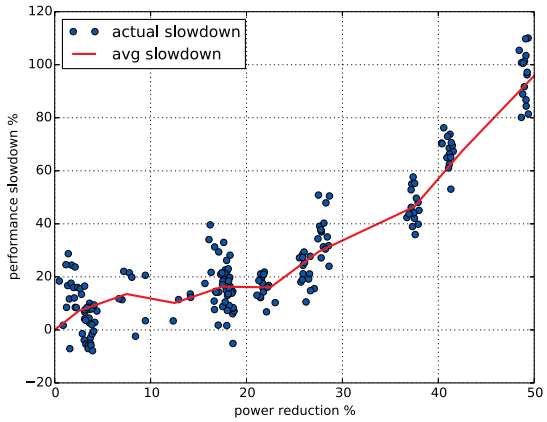

Figure 13. Web server performance slowdown at different power capping levels normalized to uncapped web servers. The y-axis is the relative performance slowdown as measured by server-side latency. The x-axis is the relative power reduction level caused by power capping.

balancing responded by sending less traffic to those servers to improve their response time during capping. To mimic a more extreme scenario where the whole cluster would be affected, we conducted an experiment with a small control group of six web servers. Power capping was applied to one group of three servers with different capping levels, while we compare their performance (server-side latency) to the other group of three servers. The results are shown in Figure 13. We see the performance decreases slowly within the 20% power reduction range. However when power capping is beyond 20%, the performance decreases faster, which may indicate that CPU frequency becomes a bottleneck.

B. Dynamic Power Oversubscription

The above two examples demonstrated how Dynamo can provide automatic protection against unexpected power surges. Next, we show an example of how Dynamo enables performance boosting via aggressive dynamic power oversubscription for a large production Hadoop cluster at Facebook.

This Hadoop cluster is located in Facebook's Prineville, Oregon data center, and spans several thousand servers. Performance tests conducted on these servers showed that turning on hardware over-clocking (Turbo Boost [25]) could improve their performance by around 13% while also increasing their power consumption by about 20%. Power planning for this cluster did not account for Turbo Boost[3], so there was no planned power margin for enabling Turbo Boost in this cluster.

In the absence of Dynamo, this would mean that we could not safely enable Turbo Boost for this Hadoop cluster, since worst-case peak power would exceed the power limit (even though average-case power would likely lie within the limit due to multiplexing [1]). However, with the power safety net the Dyanamo provides, we can oversubscribe power safely, enabling us to turn on Turbo Boost for all servers in the Hadoop cluster. This is a good example of *dynamic power oversubscription* – the servers were able to take advantage of overclocking whenever there happened to be power margin available.

Figure 14 shows the power consumption of one of the SBs connecting to this Hadoop cluster over a 24-hour period. It also shows the number of servers being capped during the same time period. We see that with Turbo Boost enabled for all servers, the SB power consumption stayed close to – but below – its 1250 KW power limit. Within 24 hours, power capping was triggered seven times, with each time lasting from 10 minutes to 2 hours, and each time throttling 600 to 900 servers slightly. Note for this case, unlike the previous two examples, the power capping was *intentional*, and is a trade-off made to improve performance by 13%.

C. Workload-Aware Power Capping

Next, we take a closer look at how Dynamo sets the power cap for servers running different services. We pick a leaf

[3]This was due to two reasons. First, a Hadoop cluster needs a large enough memory and storage footprint to operate on large data sets. This sometimes forces a lower power budget in order to squeeze in more servers into a cluster. Second, when this cluster was planned, CPU was not a major bottleneck for Facebook workloads and it was not clear whether Turbo Boost would help at that time.

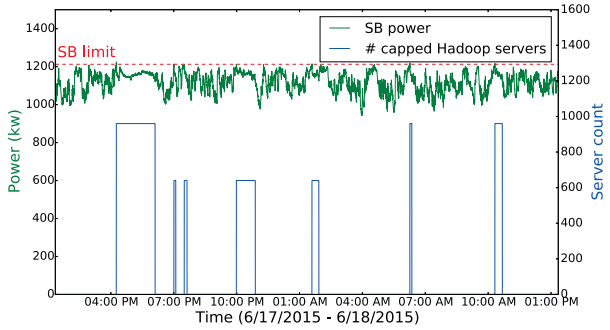

Figure 14. An example showing Dynamo-enabled performance boosting in a production Hadoop cluster in our Prineville, Oregon data center. The top line is the SB power consumption with Turbo Boost enabled for Hadoop servers. The bottom line is the number of servers being capped by Dynamo. The data spans 24 hours.

controller covering a row (RPP) of servers running a mix of services. There are around 200 front-end web servers, 200 cache servers [23], and 40 news feed servers. As discussed in Section III-C3, the leaf controller decides on optimal capping actions using priority groups. In our case, cache servers belong to a higher priority group than web and feed servers. For our experiment, we manually trigger the power capping by lowering the capping threshold during the test. Figure 15 shows the results of this experiment. The upper graph shows total row power, while the lower graph shows power breakdown by service. We see that power capping was effective between 1:50 PM and 2:02 PM. Notice that, while web servers and news feed servers were capped, cache servers were left uncapped.

To further examine the power cap settings within a priority group, we show a snapshot of the current power consumption and computed power cap value for each server in Figure 16. Recall from Section III-C3 that Dynamo uses a high-bucket-first algorithm to distribute the power cut among servers within the same priority group. For this case, the bucket is chosen to be [210 W, 300 W]. As a result, Figure 16 shows that the total-power-cut has been distributed among all web servers or feed servers whose current power consumption is 210 W or more. As expected, no cache servers were capped as they belong to a higher priority group. In the figure, the ◊ points are server power and the □ points are the power cap values (which are computed as the current power value less the calculated power cut). The algorithm also ensures that the power cap value is at least 210 W for this case.

D. Summary of Benefits

We end this section with a summary of Dynamo's benefits, as shown in Table I. Specifically:

Dynamo has prevented 18 potential power outages in the past 6 months due to unexpected power surges, similar to the case study described in Section IV-A. This happened at various levels, including RPP, PDU Breaker, and SB.

Dynamo enables us to aggressively boost performance for services via Turbo over-clocking in legacy clusters with low power budgeting. Section IV-B has already shown a Hadoop case with execution time reduced by 13% for CPU-intensive Hadoop tasks. Here we explain another example for a production search cluster at Facebook. For this search cluster, in order to accommodate more servers to get the needed memory or storage footprint to operate on large data

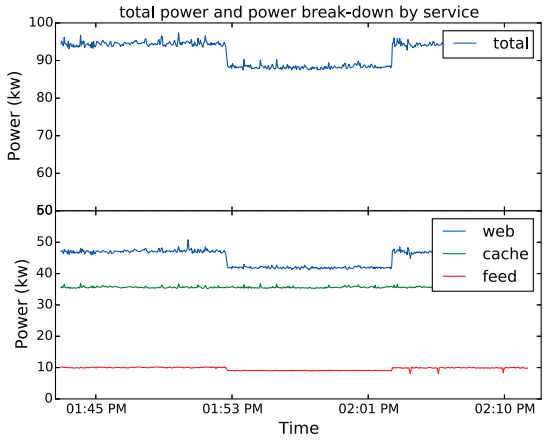

Figure 15. An example showing power capping actions for servers running different workloads or services. A leaf controller triggered power capping for a row of servers. The line in the upper-half of the graph is the total power of a row, while the lines in the lower-half of the graph are power break-down for different services, namely web, cache, and news feed.

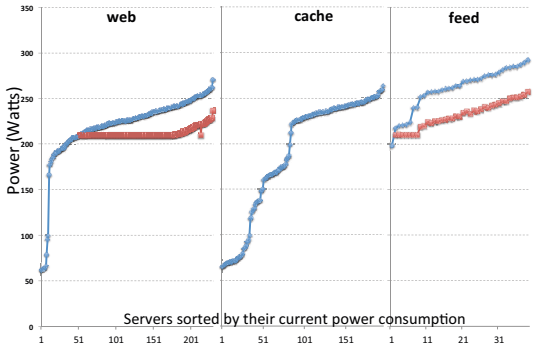

Figure 16. A snapshot of the current power consumption and the computed power cap value for each server in three service groups (web, cache, or news feed). Each ◊ point is the power consumption for a server, while each □ point is the computed power cap value for that server. The x-axis is the server index sorted by the current power consumption.

sets, all servers in this cluster were required to limit their clock frequency to make sure the worst-case application peak power is within the limited power budget. This adversely affected search performance in terms of maximum queries per second (QPS). With Dynamo, we were able to remove the clock frequency constraint and reset it to the nominal clock frequency. Further, since search is CPU-bound, we enabled Turbo Boost over-clocking for this cluster. Though this put the worst-case cluster power above its breaker limit and caused Dynamo to kick-in in rare cases, the cluster was within its power limit most of the time, and has seen up to a 40% performance improvement.

With Dynamo as a safety net, we are able to more aggressively over-subscribe our existing data centers and free up many megawatts of stranded power. As a example, we have been able to accommodate 8% more servers for the same power delivery limits in one of our data centers.

V. Related Work

To the best of our knowledge, Dynamo is the first data center-wide power management system in a real production

Use Case	Benefits
Prevent potential power outage	18 times in past 6 months
Enable performance boost for Hadoop	Boost map-reduce performance by up to 13%
Enable performance boost for Search	Boost search QPS by up to 40%
Data center over-subscription	Accommodate 8% more servers
Fine-grained real-time monitoring	3-second granularity power readings and break-down

Table I. A summary of the benefits provided by Dynamo.

environment to be described in the literature. Next, we summarize briefly other related works (which we have qualitatively compared Dynamo to throughout the paper).

There is a large body of previous work on server-level or local ensemble-level power management [4], [8], [9], [10], [26], [27], [28], [29], [30], [31], [32], [33]. Based on their control goals, they fall into the following groups: peak power management approaches [4], [8], [26]; thermal management approaches [9], [10], [27]; and average power or energy management approaches [28], [29], [30], [32], [33]. In terms of methodology, most of these works leverage the hardware P-States [11] or voltage and frequency steps (i.e., DVFS) to reach their control goals. These previous works are related to ours in the sense that they provide the foundation for the power control mechanism used in this paper. For example, Dynamo agent uses RAPL to set the power cap for individual servers, and RAPL is essentially based on DVFS and other control means discussed in this body of previous work. What distinguishes our work, however, is that we focus on data center-wide peak power management, which considers the entire power hierarchy and make coordinated control decisions.

Early work on data center-wide power management such as [34] focused on energy management which is orthogonal to our work and focus. More recent work by [12], [13] look into issues of data center-wide peak power management. The two works by [12], [13] are mostly related to our work in the sense that we have a similar focus and problem definition. However, the above two studies are either based on pure simulation or based on a small test-bed of less than 10 servers. Their proposed techniques and design do not take into account many key issues (such as power variation and breaker characterization, scalable communication between controller and controllee, workload-aware capping actions, and coordination of multiple controller instances), which need to be considered for the realistic setup of large scale data centers. In contrast, we develop Dynamo in a real production environment. The proposed techniques and design have been evaluated in very large scale data centers serving billions of users. We also report a richer set of results from a real production environment.

The work by Fan et al. at Google [1] was also in the context of a real production environment (with a very large scale data center setup similar to this study). However, their work was measurement-focused, aimed at quantifying the conservatism of nameplate-rating–based power planning. They also use simulation to analyze the potential benefits of using DVFS or other power capping mechanisms, but do not address the design of a data center-wide peak power management system.

The power variation study in Section II-B of this paper was used to understand the design space for Dynamo. One similar work we are aware of is from [22]. The methodology in [22] is similar ours in terms of using the CDF, power slope, and time window. However, [22] shows only one result at a cluster level for a fixed time window of 10 s. Our study is more comprehensive with fine-grained power samples from tens of thousands of servers for over six months. We provide characterizations for different levels of aggregation (from racks to MSBs) and at different time scales (from a few seconds to tens of minutes). In addition, we show how the variation was affected by the mix of services, which was not studied in prior work.

VI. Lessons Learned

Our experience with developing and running Dynamo over the past three years has taught us a number of lessons, which we share here.

Monitoring is as important as capping. Monitoring is critical in uncovering and preventing power issues. Many power problems we had in the past could have been avoided if we had close power monitoring to catch bottlenecks early. As a result, we have invested a lot of effort into collecting power information (e.g., fine-grained power readings from each server), and on building monitoring and automated alerting tools.

Service-aware system design simplifies capping testing. Testing power capping in production is challenging because it has the potential to negatively affect service performance. Yet, it is important to perform periodic end-to-end testing of Dynamo to ensure correctness and reliability. Our service-aware system design has helped us to address this challenge. We pre-select a group of non-critical services for the end-to-end testing of all service-agnostic logic (capping and uncapping functionality, three-band algorithm, etc.). For service-specific logic (such as service-aware optimal control actions), we use a dry-run mode and detailed logging to inspect the control logic step-by-step without actually throttling the servers in those critical services.

Design capping systems in a hardware-agnostic way. Heterogeneity in server hardware is a reality in real large-scale data centers. With rolling server generation life cycles, servers of Westmere, Sandybridge, Ivybridge, Haswell as well as Broadwell models currently coexist in Facebook data centers. And some of these platforms have individual ways to read/cap power. To reduce the system maintenance overhead, Dynamo abstracts its software into two layers – platform independent and platform-specific – and maintains the key logic transparent to platform through careful software design and abstraction. This also makes the system more scalable as it is easy to add new platforms without affecting the overall logic.

Use accurate estimation for missing power information. In a large-scale data center, hardware failures are inevitable and expected. In addition, there are planned events such as server decommissions, upgrades, and so on. All these will lead to missing server power information. We designed Dynamo to be resilient to these failures and function accurately. One method is to use system meta-data and accurate estimation for server power reading failures. Another useful tool is to use the (coarse-grained) power readings from the power breaker to validate and dynamically tune the server power estimation and aggregation. With this design, Dynamo performs accurate capping in spite of server power reading failures.

Keep the design simple to achieve reliability at scale. Dynamo agents run on every server in our fleet. As such, even a small bug can have a massive impact on the service. Staying reliable is thus a top priority for the development and deployment of Dynamo. We take a two-fold approach. First, keeping the agent and control logic simple allows us to easily identify issues when things go wrong. Second, we use a four-phase staged roll-out for new changes to the agent or control logic, so any serious issues will be captured in early phases before going wide. As a fact, Dynamo has not caused any big reliability issues in the past three years.

VII. Conclusions

Power is a scarce – and capacity-limiting – resource in data centers. Yet it is frequently underutilized due to conservative power planning. To enable better power utilization, data centers need a reliable, systemic power monitoring and management system such as Dynamo, that can safeguard against power outages with minimal impact on service performance. We have discussed Dynamo's design, and how it addresses production-environment realities such as sub-minute-granularity power variations. We then saw Dynamo in action: its intervention has helped prevent service outages in production, it has enabled dynamic power oversubscription for performance boosting, and it has allowed us to pack more servers in our data centers. Dynamo's power monitoring framework has given us fine-granularity insight into our power usage and efficiency, and we believe that there remains much room for improvement (e.g., new capping algorithms or new types of emergency response actions). With Dynamo guaranteeing power safety, we are able to experiment with more aggressive power subscription, and continue to move toward optimal data center power efficiency.

Acknowledgments

We would like to thank many Facebook engineers who have contributed to the Dynamo project, in particular Sachin Kadloor, Jiacheng Feng, Elliott Sims, Manish Mody, Yusuf Abdulghani, Parth Malani, Anand Parikh, and Bill Jia. We are also grateful for the comments and feedback received from Jason Mars and the anonymous reviewers.

References

[1] X. Fan, W.-D. Weber, and L. A. Barroso, "Power Provisioning for a Warehouse-sized Computer," ISCA, 2007.
[2] W. Turner, J. Seader, and K. Brill, "Tier Classifications Define Site Infrastructure Performance," The Uptime Institute, White Paper, 2006.
[3] J. Hamilton, "Internet-scale Service Infrastructure Efficiency," ISCA Keynote, 2009.
[4] P. Ranganathan, P. Leech, D. Irwin, and J. Chase, "Ensemble-level Power Management for Dense Blade Servers," ISCA, 2006.
[5] "Datacom Equipment Power Trends and Cooling Applications," ASHRAE, http://www.ashrae.org/, 2005.
[6] S. Gorman, "Power Supply Still a Vexation for the NSA," The Baltimore Sun, 2007.
[7] X. Fu, X. Wang, and C. Lefurgy, "How Much Power Oversubscription is Safe and Allowed in Data Centers?" ICAC, 2011.
[8] C. Lefurgy, X. Wang, and M. Ware, "Server-level power control," ICAC, 2007.
[9] D. Brooks and M. Martonosi, "Dynamic Thermal Management for High-Performance Microprocessors," HPCA, 2001.
[10] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration," ISCA, 2006.
[11] K. Taylor, "Power Management States: P-States, C-States, and Package C-States," Intel Developer Zone, 2014.
[12] X. Wang, M. Chen, C. Lefurgy, and T. Keller, "SHIP: A Scalable Hierarchical Power Control Architecture for Large-Scale Data Centers," *IEEE Trans. on Parallel and Distributed Systems*, 2012.
[13] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "Power" Struggles: Coordinated Multi-level Power Management for the Data Center," ASPLOS, 2008.
[14] "Open Compute Project," http://www.opencompute.org/.
[15] S. Muralidhar, W. Lloyd, S. Roy, C. Hill, E. Lin, W. Liu, S. Pan, S. S. hankar, V. Sivakumar, L. Tang, and S. Kumar, "f4: Facebook's Warm BLOB Storage System," OSDI, 2014.
[16] M. Slee, A. Agarwal, and M. Kwiatkowski, "Thrift: Scalable cross-language services implementation," http://thrift.apache.org/, 2007.
[17] C. Isci and M. Martonosi, "Runtime Power Monitoring in High-End Processors: Methodology and Empirical Data," MICRO, 2003.
[18] "Yokogawa WT1600 Unit:," http://www.yokogawa.com/.
[19] "RAPL for the Romley Platform, White Paper," Intel Documentation Number: 495964, 2012.
[20] H. David, E. Gorbatov, U. Hanebutte, and R. Khanna, "RAPL: Memory Power Estimation and Capping," ISLPED, 2010.
[21] "Intel Intelligent Power Node Manager 2.0: External Interface Specification Using IPMI," Intel Documentation Number: 434090, 2012.
[22] A. A. Bhattacharya, D. Culler, A. Kansal, S. Govindan, and S. Sankar, "The Need for Speed and Stability in Data Center Power Capping," IGCC, 2012.
[23] N. Bronson, Z. Amsden, G. Cabrera, P. Chakka, P. Dimov, H. Ding, J. Ferris, A. Giardullo, S. Kulkarni, H. Li, M. Marchukov, D. Petrov, L. Puzar, Y. J. Song, and V. Venkataramani, "Tao: Facebook's distributed data store for the social graph," USENIX ATC, 2013.
[24] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. John Wiley & Sons, 2004.
[25] "Intel Turbo Boost Technology 2.0," Intel Inc., http://www.intel.com/.
[26] I. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Parasol and GreenSwitch: Managing Datacenters Powered by Renewable Energy," ASPLOS, 2013.
[27] K. Skadron, T. Abdelzaher, and M. R. Stan, "Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management," HPCA, 2001.
[28] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam, "Managing Server Energy and Operational Costs in Hosting Centers," SIGMETRICS, 2005.
[29] Q. Wu, V. Reddi, Y. Wu, J. Lee, D. Connors, D. Brooks, M. Martonosi, and D. Clark, "A dynamic compilation framework for controlling microprocessor energy and performance," MICRO, 2005.
[30] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An Analysis of Efficient Multi-Core Global Power Management Policies: Maximizing Performance for a Given Power Budget," MICRO, 2006.
[31] D. Meisner, C. M. Sadler, L. A. Barroso, W.-D. Weber, and T. F. Wenisch, "Power Management of Online Data-Intensive Services," ISCA, 2011.
[32] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "CoScale: Coordinating CPU and Memory System DVFS in Server Systems," MICRO, 2012.
[33] A. Vega, A. Buyuktosunoglu, H. Hanson, P. Bose, and S. Ramani, "Crank It Up or Dial It Down: Coordinated Multiprocessor Frequency and Folding Control," MICRO, 2013.
[34] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle, "Managing Energy and Server Resources in Hosting Centers," SOSP, 2001.