SprintCon: Controllable and Efficient Computational Sprinting for Data Center Servers

Wenli Zheng[†], Xiaorui Wang[‡], Yue Ma[†], Chao Li[†], Hao Lin[§], Bin Yao[†], Jianfeng Zhang[§], and Minyi Guo[†]

[†]Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
[‡]Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, USA
[§]Alibaba Group, China
[†]{zheng-wl, lichao, yaobin, guo-my}@cs.sjtu.edu.cn [‡]xwang@ece.osu.edu [†]mayue42@sjtu.edu.cn
[§]bixuan@taobao.com [§]xingdian@alibaba-inc.com

***Abstract*—Computational sprinting is an effective mechanism to temporarily boost the performance of data center servers. However, given the great effect on performance improvement, how to make the sprinting process controllable and how to maximize the sprinting efficiency have not been well discussed yet. Those can be significant problems for a data center when computational sprinting is needed for more than a few minutes, since it requires the support of energy storage, whose capacity is limited. The control and efficiency of sprinting not only involve how fast to run servers and how to allocate resources to co-running workloads, but also the impact on power overload, and how to handle the overload with circuit breakers and energy storage to ensure power safety. Different workloads can impact sprinting in different ways, and hence efficient sprinting requires workload-specific strategies.**

In this paper, we propose SprintCon to realize controllable and efficient computational sprinting for data center servers. SprintCon mainly consists of a power load allocator and two different power controllers. The allocator analyzes how to divide the power load to different power sources. The server power controller adapts the CPU cores that process batch workloads, to improve the efficiency in terms of computing, energy and cost. The UPS power controller dynamically adjusts the discharge rate of UPS energy storage to satisfy the time-varying power demand of interactive workloads, and ensure power safety. The experiment results show that compared to state-of-the-art solutions, SprintCon can achieve 6-56% better computing performance and up to 87% less demand of energy storage.

***Index Terms*—computational sprinting, power control, data center, server, power management**

I. Introduction

Computational sprinting is a mechanism proposed in recent years to temporarily boost the system performance of computers. It is originally proposed to speed up the processing of workload bursts on mobile devices [1], with the activation of normally-off cores, and the enhancement of heat dissipation and power supply. This concept has later been applied to data center operation [2], [3]. When handling workload bursts, sprinting allows servers to activate more hardware resources or run at higher frequencies, which are normally prohibited by power limits, and thus improves system performance in a short duration. The sprinting power, i.e., the increased power during a sprinting process, can significantly overload the primary power infrastructure. Hence energy storage devices (e.g., UPS batteries) are usually exploited to provide additional power, while the circuit breakers (CBs) can tolerate some overload degree, i.e., the ratio of delivered power over rated power, without tripping up or damaging power infrastructure in the sprinting duration, i.e., the time duration of a sprinting process.

An efficient sprinting process requires compromise between sprinting power and sprinting duration, to pursue the optimal computing performance of processing a workload burst. This is because the battery capacity as well as the CB tolerance of overload are necessary to support sprinting, but both of them are limited, and hence higher sprinting power leads to shorter sprinting duration. Depending on the workload demand, sometimes we may prefer relatively long sprinting duration, e.g., 15-30 minutes, to speed up processing the client requests related to a certain event. To make sure the data center servers can keep sprinting in a preset duration, we need to control the sprinting power. While traditional power control of computers usually handles workload bursts with conservative management, e.g., throttling devices for power reduction with performance degradation, computational sprinting proactively increases power supply to improve computing performance. However, previous studies on computational sprinting mainly focus on how to enable power increase, and how to determine which workloads or devices to use the sprinting power, without rigorous feedback control that enforces the sprinting duration and guarantees the power safety of a data center, which is particularly important as the sprinting power can consume the headroom in the data-center level power budget.

Controllable computational sprinting comes with three major challenges. First, how to do power control without violating performance requirement or causing additional costs. Although manipulating battery discharge power to control power load can avoid performance degradation, the limited battery capacity can be a problem for relatively long sprinting duration, as UPS batteries might be provisioned for only 5

This work was supported in part by NSFC under Grants 61702329, 61872235, 61729202, 61832017 and U1636210, in part by the National Basic Research Program (973 Program, No.2015CB352403), and in part by The National Key Research and Development Program of China (2018YFC1504504, 2016YFB0700502).

1530-2075/19/$31.00 ©2019 IEEE
DOI 10.1109/IPDPS.2019.00090

minutes in some data centers. In addition, deep discharges can significantly affect the lifetime of batteries and thus their replacement costs, if they are used multiple times a day for periodic sprinting (as suggested in [2], [4]). Second, with limited battery discharge, which devices to throttle if necessary. Sprinting power provides more computing capacity than usual, which however may not be demanded by all the devices, depending on the workloads being processed. Some types of workloads can tolerate relatively slow processing or deferment, e.g., batch workloads, whose deadlines are usually in term of hours or even days, but can become in minutes after being postponed. Such batch workloads need to run simultaneously with interactive workloads even if the latter comes with heavy load, but are still not as latency-critical as the latter. Therefore, workload-specific strategies are needed for the power control of sprinting. Third, given deadlines of batch workloads, how much to throttle their processing devices. While the actual power consumption varies dynamically, the power delivered by CBs must be controlled within a preset power budget to avoid tripping. If the CB tolerance can be efficiently utilized with appropriate dynamic throttling, it can help with executing batch workloads in time.

In this paper, we propose SprintCon, a controllable and efficient computational sprinting mechanism for data center servers. In sharp contrast to existing work, SprintCon adopts feedback control theories to control the sprinting power, to guarantee the safety of overloading circuit breakers and avoid affecting the power budget headroom, and improve computing efficiency, energy efficiency as well as cost efficiency. SprintCon mainly consists of a power allocator and two different power controllers. The allocator divides the power load to different power sources, i.e., the UPS and the primary power system protected by CBs, to exploit the CB tolerance and the flexibility of UPS. The server power controller adapts processors to handle batch workload, which allows the processing to be power-efficient and done before the deadline. The UPS power controller dynamically adjusts the discharge rate of batteries to follow the fluctuating power consumption, such that the CB overload can keep constant and safe, and the sprinting duration can also be maintained. We have evaluated SprintCon by comparing it with state-of-the-art solutions, which also utilizes CB tolerance of overload and UPS battery for computational sprinting. The experiment results show that SprintCon can provide 6-56% higher computing capacity with up to 87% less use of energy storage.

Specifically, the major contributions of this paper are:

- We propose to proactively control the power load of computational sprinting for data center servers to guarantee power safety, and improve computing efficiency, energy efficiency as well as cost efficiency.
- We propose to exploit CB tolerance of overload and energy storage capacity in different ways for sprinting power, based on their complementary advantages.
- We propose to use different control mechanisms to handle the sprinting power caused by batch and interactive workloads, based on their different demands of power.
- We have evaluated SprintCon by comparing it to state-of-the-art baselines, and shown the advantage of SprintCon in terms of computing performance and resource usage with experiment results.

The rest of the paper is organized as follows. Section II summarizes the related work. Section III explains the motivations of this work. Section IV and Section V discuss the design details of SprintCon. Sections VI and VII present the evaluation methodology and results. Section VIII concludes the paper.

II. Related Work

Computational sprinting is originally proposed by Raghavan et al. [1], [4] to temporarily boost workload processing on mobile devices. After that, Zheng et al. [3] enable this methodology in data centers, using UPS batteries and thermal energy storage to provide additional power and cooling, allowing more computing resources to be powered on to handle workload bursts. Fan et al. [2] integrate data center sprinting with game theories to intelligently do sprinting based on the dynamic states of applications and systems. Rezaei et al. [5] use distributional sprinting to speed up computing streaming applications. Yang et al. [6] identify the bottleneck service to maximally boost the performance. Cai et al. [7] exploit renewable energy for computational sprinting. While their work discusses how to enable computational sprinting, how to increase power supply, and how to determine which devices or workloads to use the increased power, SprintCon addresses how to make computational sprinting controllable for improved safety and efficiency, and integrates computational sprinting with workload-specific power control, both of which have not been studied before.

Though not integrated with sprinting, power control with minimized performance degradation has already attracted the attention of many researchers. For example, Lefurgy et al. [8] propose a power capping strategy to maximize the system performance while precisely controlling server power to stay below a given budget. Wang et al. [9], [10] have designed power control algorithms at the enclosure and data center levels. Due to the advantage of well-established control-theoretic design, those solutions can achieve much better system performance compared with open-loop or heuristic solutions. In recent years, studies have been conducted to operate more server components and data center facilities. Chen et al. [11] and Deng et al. [12] both propose power management solutions that coordinate the power states of CPU and memory, and Chen et al. [13] jointly optimize multi-core frequency scaling and workload scheduling, in order to improve the system performance and/or to save energy. Zheng et al. [14] make use of electrical and thermal energy storage devices for power shaving, flattening the power-time curve of a data center. Li et al. [15] and Hou et al. [16] both address peak power problems that can be caused by malicious users in data centers, with backup batteries and elastic pricing strategies to manipulate the power loads. Different from the existing power control solutions that conducts conservative power

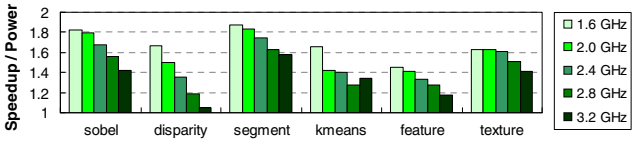


Fig. 1. Computational sprinting speeds up processing, but the per-watt speedup decreases with the increase of processor frequency in general. Y-axis coordinates indicate the ratio of speedup over normalized power consumption.

management, SprintCon supports computational sprinting for enhanced power supply and computing capacity.

III. Motivation

In this section, we explain why it is beneficial to control computational sprinting, and the motivations of this work.

First of all, it is necessary to understand why we sometimes want to constrain sprinting power, i.e., do computational sprinting with lower sprinting power but longer sprinting duration. Actually it is easier to allow high-power sprinting in a short time and then return to normal processing in the rest of time. Higher-power sprinting enables higher processor frequencies and hence higher speedup, which has been studied in [4], and Figure 1 presents the analysis of some of their results. We can observe that when running six different workloads, in general, the per-watt speedup decreases with the increase of processor frequency. It can be explained by two reasons: (1) Processors are not the only hardware that impacts computing speed. Other hardware becomes the bottleneck when processor frequency is high. (2) Processor frequency is known to be positively correlated with power, but not linearly. A little higher frequency can cause much higher power consumption. In total, it means that given a certain fixed amount of additional energy (e.g., from discharging batteries or overloading CBs), lower sprinting power is expected to achieve higher computing efficiency and energy efficiency on average during a relatively long time period, compared to higher sprinting power.

Secondly, it is important to control the power delivered by the circuit breaker to a constant power budget, in terms of power safety and energy efficiency. Figure 2 shows a typical trip time curve of circuit breaker, from which we can see the trip time is a nonlinear function of the overload degree. This means if the power delivered by CB dynamically changes, predicting when the CB would trip up is hard to be accurate. To avoid tripping up the CB for power safety, a feasible solution is to conservatively estimate the trip time based on the maximum power, and end up overloading within the estimated trip time. To maximize the utilization of CB tolerance of overload for energy efficiency, we need to minimize the difference between the maximum power and the time-varying actual power. If we proactively determine the maximum power as a constant power budget, minimizing the difference becomes a typical power control task, i.e., controlling the actual CB power to the determined power budget.

At last, power control for computational sprinting is a non-trivial work. Although power control has been extensively researched before [11], [19]–[21], previous studies mainly try

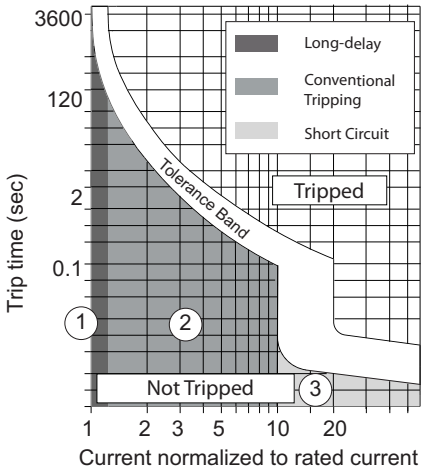


Fig. 2. The trip time curve of Bulletin 1489-A Circuit Breaker [17], [18]. The trip time is a nonlinear decreasing function of the overload.

to improve the system performance given a fixed power capacity, without exploiting the UPS batteries for enhanced power supply. In addition, existing power control research does *not* differentiate interactive and batch workloads. While interactive workloads are delay sensitive, slowing down the processing of batch workloads causes little to no revenue loss as long as their execution is completed before deadlines [22]. Therefore, power control for computational sprinting needs workload-specific strategies, such that performance requirements of different workloads can be all satisfied when we select to do computational sprinting with lower sprinting power and thus lower processor frequencies. Unfortunately, existing power control solutions (and some computational sprinting solutions) are not designed to this because they mainly aim to improve the IPS (instructions per second) performance of the entire system, and thus it becomes a challenge to our work.

The above reasons motivate us to design SprintCon for controllable computational sprinting with workload-specific strategies, in order to handle long-term workload bursts, with better computing efficiency, energy efficiency and power safety. The improvement on energy efficiency further leads to better cost efficiency, because less usage of UPS batteries can reduce the battery replacement costs. Please note that SprintCon can also do short-term sprinting, though controlled sprinting may not have much advantage over uncontrolled sprinting when the workload burst duration is short.

IV. Principles of SprintCon

In this section, we present the principles of SprintCon, including the design of power load allocator and how the power control mechanism works. In general, the allocator determines the control targets for the two controllers, and the two controllers do their respective jobs individually.

A. Control Target for UPS Power Controller

The power load allocator quantitatively determines how to supply the additional power required by computational

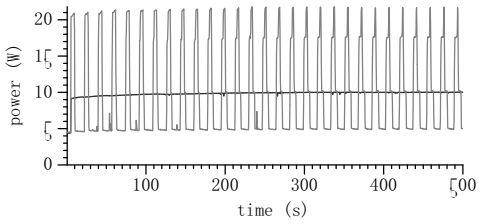


Fig. 3. An example of periodic computational sprinting [4], whose sprinting period is about 18 seconds.

sprinting, which comes from overloading the circuit breaker and discharging the UPS battery. Specifically, to divide the power load between the two power sources, the allocator determines the control target of power load on CB, and the remaining power load is left on UPS. This power control target, noted as P_{cb}, determines the CB overload degree and duration, i.e., how much and how long to overload the circuit breaker, ahead of time, based on the workload burst duration T_{burst}. When T_{burst} is short, e.g., less than 1 minute, it is perhaps unnecessary to control the sprinting power, and no need to constrain the CB overload. When T_{burst} is longer, e.g., 5-10 minutes, we can set the overload duration to equal the workload burst duration, to maximize the additional energy provided for sprinting. When T_{burst} is much longer, e.g., 15 minutes or more, periodic overload, i.e., switching CB between the normal operation state and the overload state, can be more efficient. During the normal operation state, CB can recover from being close to tripping, and get ready for the next overload.

SprintCon is designed to mainly address such long-term sprinting, for which it periodically overloads CB and adjusts the value of P_{cb}. Specifically, P_{cb} is equal to the rated capacity of CB at the normal operation state, and equal to the product of rated capacity and overload degree at the overload state. For example, if the rated capacity of CB is 3 kW and the CB overload degree is 1.3, P_{cb} will be set to 3.9 kW in the overload duration, and then adjusted to 3 kW in the recovery period, and then adjusted to 3.9 kW again, repeating periodically. Although periodic CB overload can naturally support periodic sprinting, like what is shown in Figure 3 [4], it does not satisfy the requirement of long-term sprinting. When CB is recovering, to supply sprinting power, especially for the interactive workloads that demand of more power to speed up, SprintCon exploits UPS batteries to cover the gap between the power supplied via CB and the power demand, such that the interactive workloads can have adequate power for fast execution. The UPS discharge power is dynamically manipulated by the UPS power controller of SprintCon, such that the power delivered through CB can be controlled to P_{cb}.

B. Control Target for Server Power Controller

While the UPS power controller provides the power safety of overloading CB and the computing efficiency of interactive workloads, the server power controller improves the energy efficiency while completing the execution of batch workloads before their deadlines. Both overloading CB and discharging UPS can supply sprinting power, but the latter uses stored backup energy, leading to risks of higher battery replacement costs and lower data center availability. Therefore, SprintCon constrains the power consumption of batch workloads, i.e., running delay-insensitive batch workloads at appropriately low frequencies, to lower the total sprinting power, and hence reduce the demand of UPS discharge. Intuitively, the optimal operation is to keep the total power consumption of all the workloads as close to P_{cb} as possible, such that the UPS discharge is minimized.

However, at the rack level the amount of interactive workload can fluctuate dramatically and frequently, while it takes time for the controlled power to converge. The interactive workloads usually come from interactive or streaming applications, and servers need to process them with peak frequencies during a workload burst. Hence their power consumption mainly fluctuates with the amount of workload. Consequently, controlling the total power is hard to achieve satisfactory control accuracy, and might even affect the system stability. Therefore, in addition to setting P_{cb}, the power load allocator determines another power control target, which is used by the server power controller to control the batch workload processing power. This target, noted as P_{batch}, is usually less than or equal to P_{cb}. If P_{batch} is less than P_{cb}, the remaining amount can be used by processing interactive workloads.

The power load allocator determines P_{batch} based on two factors: (1) The progress of batch workload processing and the remaining time before its deadline. We consider the execution time of a batch workload can be estimated based on short-term profiling, which collects the statistics of used CPU cycles and cache misses in milliseconds. While the server power control dynamically manipulates the processor frequencies, the power load allocator uses the progress model in [12] to calculate the impact of processor frequency scaling on the execution time. If the remaining execution time of some batch workloads is expected to violate their deadlines, the value of P_{batch} would be increased. (2) The fluctuation of interactive workload power consumption. When most of the time, e.g., more than 90% of the time, the interactive workload power consumption is higher than P_{cb} - P_{batch}, it means the power capacity provided by overloading CB is highly utilized, and thus P_{batch} can be decreased to make interactive workloads use more CB power instead of UPS power. If the interactive workload cannot sufficiently utilize the CB power capacity, P_{batch} would be decreased to allow batch workloads use more power. Based on the two factors, the power load allocator dynamically adjusts P_{batch} but with a much longer operation period than the settling time of the server power controller, e.g., 30 seconds, such that the controlled batch workload power consumption can converge to P_{batch} before it is adjusted again.

C. Power Control Mechanism

Figure 4 shows the general constitution diagram of SprintCon, whose core algorithm is implemented in software but can interact with the hardware components. During the sprinting

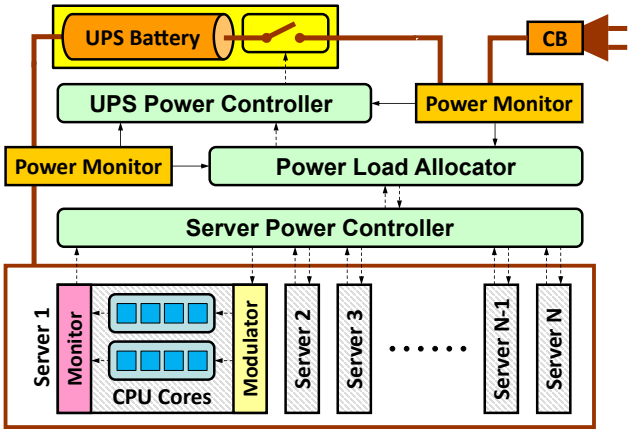


Fig. 4. General illustration of SprintCon constitution, and how SprintCon works with the servers and power infrastructure.

process, SprintCon uses two power controllers to manipulate the servers and UPS energy storage. For batch workload, SprintCon controls their power consumption to P_{batch}. For interactive workload, SprintCon controls the discharge power of UPS, such that the actual CB power is maintained within P_{cb}. The states of circuit breaker and energy storage need to be monitored during the sprinting process. If the circuit breaker is close to tripping, SprintCon stops overloading it and makes the energy storage to take over the rest power load. If the energy storage is running out, SprintCon makes the P_{cb} as the power control target for the power consumption of all workloads. In this situation, the power budget may be inadequate, and different workloads can bid for power as in [2]. If both the two events occur, SprintCon ends sprinting. The power control mechanism of SprintCon works in the following steps.

The server power controller is designed to improve the computing efficiency and energy efficiency, with workload-specific strategies. During a sprinting process, it keeps all the CPU cores processing interactive workloads at their peak frequencies, and dynamically adapt the cores running batch workloads for power control. Specifically, after the power load allocator determines the power control target P_{batch}, the following feedback control loop is invoked at the end of every control period:

1) The server monitors report the utilization of each CPU core (e.g., with performance counters) that runs batch workload in the last control period to the controller;
2) Based on the collected CPU utilization data and the power budget P_{batch}, the controller calculates and outputs the new frequencies of those CPU cores in the next control period;
3) The server modulators adjust the frequencies of CPU cores (e.g., with writing system files) according to the controller outputs;
4) The controller checks with the power load allocator whether P_{batch} is changed, and if so, updates this power control target from next control period.

The server power controller is designed with the ability to manipulate all the processors on the rack of servers, but most of the time it only dynamically adapts the cores running batch workloads. In modern data centers, interactive and batch workloads can be processed on different servers to avoid interference, or on the same server to improve utilization [23]. SprintCon can deal with both the two situations, though P_{batch} cannot be directly measured with a typical power monitor in the latter situation. To solve this problem, we model the feedback power measurement in Section V-A.

The UPS power controller is used dynamically to manipulate the UPS discharge power, in order to control the actual power load on CB to the control target P_{cb}, and hence guarantee the power safety. In every control period, a power monitor reports the total power consumption of the server rack, p_{total}, and the UPS discharge power needs to be equal to p_{total} - P_{cb}, or zero if $p_{total} < P_{cb}$. Quantitatively and dynamically configuring the UPS discharge power can be realized by using the Charge / Discharge Circuit Design in [24] and managing the on/off states of its switches with duty cycling control. For example, the UPS power controller can set the duty ratio at $x\%$ to make the UPS discharge $x\%$ of total power consumption.

D. Hardware Adaptation Methodologies

We adopt well-established methodologies to enable the adaptation of server hardware components. To dynamically adapt the frequency of each CPU core, we use dynamic voltage and frequency scaling (DVFS) due to its small overheads and fast response time. Since the focus of this paper is not about parallel processing of multi-thread applications, we assume the workload processed by each core is independent of the others for simplicity. If the parallel processing issue needs to be addressed, we can integrate SprintCon with existing chip-level power allocation strategies, e.g., [25]–[28], making SprintCon to determine the total frequency quota of a group of cores running the same application, and then divide the frequency quota to the cores in the group.

V. Server Power Controller

In this section, we first introduce the power models of our target system, and then describe how we design the server power controller in detail.

A. Power Modeling

Although the CPU power is a cubic function of processor frequency when DVFS is activated, previous studies have shown that the server power consumption can be an approximately linear function of processor frequency [8], [29]. This is reasonable because when the processor frequency is relatively low and bottlenecks the throughput, both the CPU power and non-CPU power increase with the increase of processor frequency, since the non-CPU hardware can be more utilized when throughput increases; but when the processor frequency is relatively high and non-CPU hardware becomes the bottleneck, the increase of processor frequency only increases the CPU power. We adopt the linear model from

[29] to calculate the power consumption of processing batch workloads on server i as follows, based on core frequencies:

$$p_i(t) = k_i \sum_{j=1}^{m_i(t)} f_{i,j}(t) + c_i m_i(t)/M_i \quad (1)$$

where $f_{i,j}$ is the frequency of core j on server i, k_i is the slope parameter of the linear function, and c_i is the frequency-independent part of server power. The server has M_i cores and $m_i(t)$ of them are processing batch workloads at time t. Since a batch workload usually takes a relatively long time to execute, $m_i(t)$ can be regarded as unchanged before the number of batch workloads that are being executed changes. Therefore, Equation (1) can be simplified to:

$$p_i(t) = K_i f_i(t) + C_i \quad (2)$$

where $K_i = m_i k_i$, $C_i = c_i m_i / M_i$, and f_i is the average frequency of the cores running batch workloads on server i. Based on the power model of processing batch workloads on a server, we can model the total power consumption of processing batch workloads on a rack of servers, p_{batch}. With $\mathbf{K} = [K_1, K_2, ..., K_N]$ and $\mathbf{F} = [f_1, f_2, ..., f_N]^T$ for a rack of N servers, we have the matrix form for p_{batch}:

$$p_{batch}(t) = \mathbf{K}\mathbf{F}(t) + C \quad (3)$$

where C is the sum of C_i for $i = 1, 2, ..., N$. While the processor power model in [29] is also related to utilization, here we assume constant utilization to make the power linearly related to the frequency, which is acceptable because the server power controller of SprintCon is designed to tolerate some modeling errors (such as the variation of utilization), as explained in Section V-C. With $\Delta\mathbf{F}(t) = \mathbf{F}(t+1) - \mathbf{F}(t)$, we derive the power model as a difference equation:

$$p_{batch}(t+1) = p_{batch}(t) + \mathbf{K}\Delta\mathbf{F}(t) \quad (4)$$

On the other hand, we model the power consumption of processing interactive workloads as a linear function of CPU core utilization [29], because SprintCon keeps running those cores at their peak frequencies during the sprinting process:

$$p_{inter}(t) = \mathbf{K}'\mathbf{U}(t) + C' \quad (5)$$

where $\mathbf{U} = [u_1, u_2, ..., u_N]^T$, and u_i is the average processor utilization of the CPU cores running interactive workloads on server i. Although SprintCon does not control p_{inter}, we still model it because it is used to calculate the feedback power consumption of processing batch workloads:

$$p_{fb}(t) = p_{total}(t) - p_{inter} \quad (6)$$

where p_{total} is the total power consumption of the server rack, and it can be physically measured by a power monitor. p_{inter} is calculated based on Equation (5), whose input variable u_i can be also physically measured by the processor utilization monitors on servers. Therefore, considering we cannot directly measure p_{fb}, we use Equation (6) to derive its value as the feedback input for the server power controller.

While we model the power consumption with respect to the server hardware, in practice there are other devices on the rack that can impact the power variation, e.g., the cooling fans, which can be installed on each server or shared by multiple servers. The fan power consumption depends on not only the server power consumption, but also the temperature set point and the ambient air temperature, leading to system errors that need to be handled by feedback control. Such factors that are difficult to be accurately modeled highlight the significance of applying feedback control in data center power management, which are not well addressed in previous computational sprinting solutions.

B. Control Algorithm

The server power controller manages the frequencies of the CPU cores that process batch workloads, and controls $p_{batch}(t)$ to the target P_{batch}. It is designed based on the *Model Predictive Control* (MPC) [30] theory, which is an advanced optimal control technique and suits our problem well, because it can deal with a MIMO control problem with specified constraints and theoretically guarantee the system stability. In a stable MPC-controlled system, the real power consumption can converge to its set point as long as the inputs are within bounded ranges.

The MPC algorithm optimizes a cost function periodically, to minimize the difference between a series of predicted controlled variables and a reference trajectory. The length of the series, L_p, is called the prediction horizon. The controlled variables are predicted based on the current operation, assuming the same operation will continue in the following L_p control periods. For example, if the current operation is to reduce the CPU frequency by 100 MHz, then the CPU frequency is assumed to be further reduced by 100 MHz in each control period. In our problem, the controlled variable is p_{batch}, the total power consumption of processing batch workloads, and it is designed to trace the following reference trajectory p_r to approach P_{batch}, the power budget determined by the power load allocator.

$$p_r(t+x|t) = P_{batch} - e^{-\frac{\tau}{\tau_r}x}(P_{batch} - p_{fb}(t)) \quad (7)$$

With a larger time constant τ_r, the system can have a smaller overshoot but may converge slower to the set point. In addition, our algorithm also minimizes the difference between the manipulated variables (i.e., the CPU core frequencies) and their optimal values (i.e., the peak frequencies in our scenario). We also consider a series of manipulated variables with a length of L_c (called the control horizon). Based on those principles, the cost function at time t can be written as:

$$W(t) = \sum_{n=1}^{L_p} \|p_{batch}(t+n|t) - p_r(t+n|t)\|^2_{Q(n)} + \sum_{n=0}^{L_c-1} \|\Delta\mathbf{F}(t+n|t) + \mathbf{F}(t+n|t) - \mathbf{F}_{max}\|^2_{\mathbf{R}(n)} \quad (8)$$

where F_{max} is the vector that contains the peak processor frequencies. In the cost function, the first term indicates the difference between the controlled power consumption p_{batch} and the reference trajectory p_r (i.e., tracking error), and the second term indicates the differences between the actual processor frequency and the peak frequency (i.e., control penalty). By minimizing the tracking error and control penalty, SprintCon

can make the power consumption converge to its set point while optimizing the overall system performance. Q and $\mathbf{R}$ are the weights for tracking error and control penalty, respectively. In every control period, the cost function is minimized subject to:

$$F_{i,min} \leq f_{i,j}(t) \leq F_{i,max},\ j \in [1, m_i(t)],\ i \in [1, N] \qquad (9)$$

This constraint sets the lower and upper bounds of the manipulated variables, i.e., the processor frequencies. Please note that the feedback variable $p_{fb}(t)$ (i.e., the feedback power consumption) in Equation (7) needs to be updated in every control period, such that the reference can be dynamically adjusted according to the feedback, to correct the control errors caused by the inaccurate prediction, because the power models have some errors inevitably (e.g., affected by the processor utilization).

The control penalty weight $\mathbf{R}$ in Equation (8) significantly impacts the power allocation among different CPU cores. The server power controller manipulates the value of $\mathbf{R}$ to balance the execution progresses of different batch workloads. Given $\mathbf{R}$ as a vector of $R_{i,j}$ for i = 1, 2, ..., N and j = 1, 2, ..., m_i, we have $R_{i,j}$ equal to the ratio of remaining execution progress over normalized remaining time before deadline. For example, if a batch workload running on the jth core of server i has been executed for 80%, used 6 minutes and had 4 minutes left before its deadline, then $R_{i,j}$ = (1 - 80%) / (4 / (6+4)) = 0.5. In this way, the batch workload with less execution progress and/or less remaining time can get more power to speed up.

C. Stability

While the power models used to design the server power controller are linearized with respect to the CPU core frequencies, on a real server rack there exist various factors that can cause errors of system modeling, e.g., the processor utilization and the cooling fan power consumption. An important reason for us to adopt a control-theoretic approach like MPC is that it can provide the theoretical guarantee for system stability, as long as the modeling errors stay within allowed ranges. In each control period t, the MPC algorithm solves an optimization problem (Equation (8)) to obtain the optimal solution $\Delta\mathbf{F}(t)$, which determines the manipulation of processor frequencies in the tth period. Based on $\Delta\mathbf{F}(t)$ and the system gain matrix that characterizes the modeling errors, we can derive the close-loop system model. If all the poles of the close-loop system are located within the unit circle of the complex space, the actually controlled system is theoretically guaranteed to be stable. Our analysis shows that the system stability can be guaranteed for the scenario of server rack power control, though the details are not presented here due to space limitations. As the stability of the server power controller is theoretically guaranteed, at the higher-level of the control framework, we configure the power load allocator to adjust the power control target P_{batch} with a longer period than the settling time of the server power controller. In this way, the power control target will not change before the power consumption p_{batch} converges to the power budget.

VI. Evaluation Methodology

To evaluate our design, we set up the experiment environment as a combination of physical tests and simulation. SprintCon is compared with state-of-the-art solutions for computational sprinting in data centers.

A. Experiment Setup

In the physical tests, we use two PowerEdge 730 servers to run interactive and batch workloads. The interactive workloads are generated based on the traces from a Wikipedia data center [31]. For the batch workload, we run SPEC CPU2006 benchmarks (CINT2006 benchmarks 400, 401, 403 and 429 on one server; CFP2006 benchmarks 433, 444, 447 and 450 on the other server). We execute four workloads on the four cores of each server, and use performance counters to collect the execution data (e.g., used CPU cycles and cache misses) to form 15-minute workload traces. The traces from Wikipedia are longer than 15 minutes and hence long enough to generate the 15-minute execution data traces for interactive workloads; the batch workloads are processed repeatedly and continuously, i.e., when a batch workload is completely executed in less than 15 minutes, it is re-executed immediately, until the workload is run for 15 minutes. Those workload traces are then applied in a simulated server group. Each server is equipped with two 4-core CPUs and the processor frequency ranges from 400 MHz to 2.0 GHz.

In order to highlight that SprintCon is robust and can tolerate the modeling errors, we use more accurate models for power measurement in the simulation than the linear power models used in the controller design. To simulate the measurement of server power, we use the power model from [29], which takes both frequency and utilization as inputs. The simulation runs 16 servers in total. Each of them consumes 150 Watts when being idle and 300 Watts when being fully loaded and run at the peak frequency. Therefore, the total power consumption can be at most 4.8 kW. The UPS battery capacity is configured to supply the maximum total power for 5 minutes, and hence the energy capacity is 400 Wh. To simulate a server rack with power-oversubscription that save the capital expenses for data centers, we set the rated power capacity of the circuit breaker at 3.2 kW, i.e., 2/3 of the maximum total power. For the CB overload degree and duration, we set them to be 1.25 and 150 seconds, and the CB takes no more than 300 seconds to recover from being close to tripping. The three numbers are kept the same as those in [2], since we use their solution for comparison in our evaluation.

B. Baseline Solutions

To evaluate SprintCon, we compare it with the state-of-the-art baselines from [2]. However, while their work mainly proposes sprinting game with their Equilibrium Threshold solution, it is designed for a multi-user scenario and hence not appropriate for the comparison with SprintCon. Instead, we select to run the sprinting game with their Cooperative Threshold solution, which maximizes system performance like what SprintCon does. We name this baseline solution as SGCT.

The major difference in the targets of SGCT and SprintCon is that SprintCon maximizes the performance of interactive workloads, but SGCT maximizes the performance of all the workloads.

In addition, we make a variant of SGCT and call it SGCT-V1, because SGCT can trip circuit breakers and cause significant degradation in energy efficiency. SGCT-V1 solves this problem by ideally manage the processor frequency and power consumption of servers, such that the circuit breaker is exactly overloaded at the present target, i.e., P_{cb}, though this is not feasible in practice without closed-loop control. In this ideal way, SGCT-V1 can avoid trip the circuit breaker.

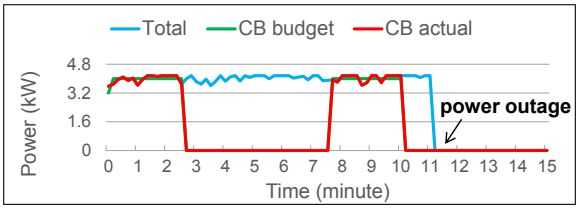
Due to the differences in studied scenario, here we use the processor utilization as the metric to determine which workload (or actually which CPU core) to sprint for the sprinting game. A CPU core with higher utilization is considered to have a higher demand of the computing resource, and hence we allow it to run at the peak frequency. Considering this customization can make SGCT and SGCT-V1 less efficient for processing interactive workloads, whose processor utilization is typically lower than that of batch workloads, we generate another variant of SGCT and call it SGCT-V2. It differs from SGCT-V1 by giving a higher priority to sprint the CPU cores running interactive workloads rather than the cores running batch workloads.

VII. Results

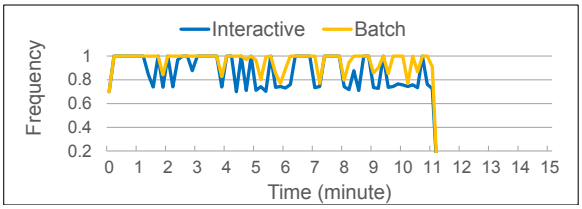
In this section, we present the experiment results and compare SprintCon with the state-of-the-art baseline solutions, SGCT, SGCT-V1 and SGCT-V2.

A. Uncontrolled Computational Sprinting

Power safety and availability are the most critical issues for data center power management. However, to achieve better computing performance, computational sprinting usually causes additional power load and discharges backup energy storage capacity, which can easily lead to problems of safety and availability. Here we first demonstrate the potential risks due to uncontrolled computational sprinting with experiment results. As shown in Figure 5(a), SGCT does not rigorously control the sprinting power to the power budget (the CB actual power can be slightly higher than the CB budget power), and at the same time it uses overloading CB as the only knob to provide sprinting power at the beginning. Hence SGCT can trip the circuit breaker in 150 seconds, and after then it uses UPS as the only power source for the entire server rack. Such a large power load draws the UPS battery capacity quickly, and in the second CB recovery period, the UPS runs out of stored energy after the 11th minute. Moreover, since the CB has not recovered yet, the servers have no power supply at all, and thus can be only shutdown. This is why in Figure 5(b) we can find the processor frequencies for both interactive and batch workloads drop down to out of the scope (actually become zero). From the above results, we can see that uncontrolled computational sprinting not only risks to trip up circuit breakers and causes problems in safety and availability,



(a) Power Curves



(b) Average frequency=0.64 for Interactive & 0.71 for Batch

Fig. 5. Uncontrolled computational sprinting can easily trip up circuit breakers, and hence load the UPS with all the server power demand. In this way, the UPS battery capacity can run out in a few minutes, which not only disables long-term sprinting, but also leads to power outage if the circuit breakers cannot recover in time.

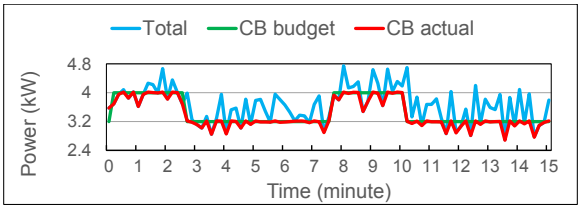
but can also consume the UPS energy capacity in an inefficient way, and hence disables long-term sprinting.

B. Comparison on Power Behavior

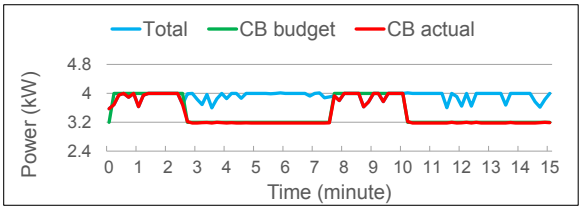
SprintCon not only controls the sprinting power, but also controls it in an energy-efficient way. Figure 6(a) shows that SprintCon highly utilizes the CB tolerance of power overload and regards it as the major energy source for sprinting power. Relatively, its usage of UPS energy, which is the gap between the Total power curve and the CB actual power curve, is significantly less. The fluctuation of the Total power curve is mainly caused by the time-varying amount of interactive workload, from which we can see SprintCon exploits the flexibility of UPS discharge power adjustment with the UPS power controller. In contrast, the Total power curves for SGCT-V1 and SGCT-V2 are nearly flat (see Figures 6(b) and (c)), because they primarily keep the total budget of sprinting power unchanged unless the workloads do not need so much power. These two baseline solutions use UPS batteries as a backup power source and only discharge UPS after the CB can no longer be overloaded (until the next CB overload period). Compared to SprintCon, SGCT-V1 and SGCT-V2 consume less power when the CB is overloaded, but more power when the CB is recovering.

C. Comparison on Frequency Behavior

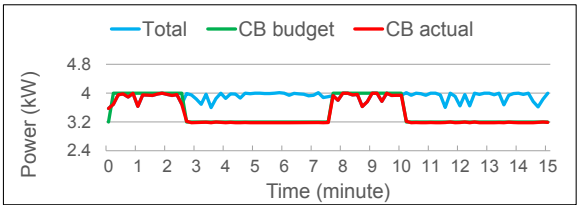
Figures 7(a), (b) and (c) show the comparison of SprintCon, SGCT-V1 and SGCT-V2 on computing capacity, or specifically, on the average frequencies of processing interactive and batch workloads, respectively. It is clear that SprintCon keeps processing interactive workloads at the peak frequency and dynamically adapts the processor cores running batch workloads. When more sprinting power is available, the batch workloads can use more power and get run with higher frequencies. With limited sprinting power, the batch workloads can be



(a) SprintCon

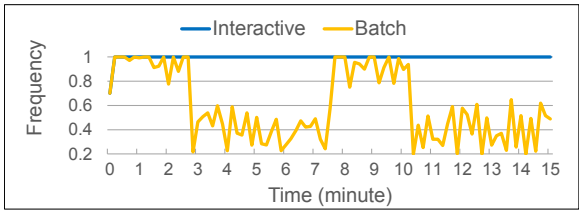


(b) SGCT-V1

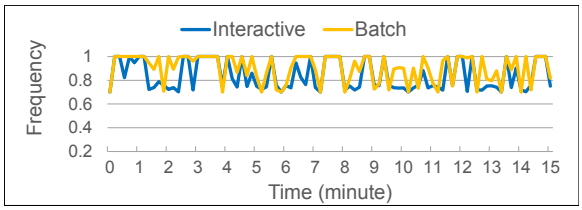


(c) SGCT-V2

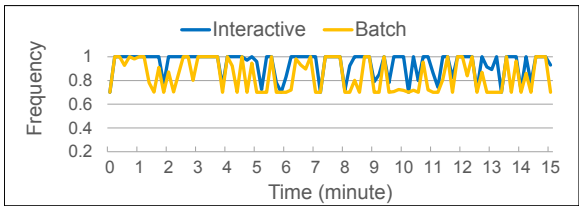
Fig. 6. SprintCon controls the power load on the circuit breaker to ensure it would not trip up, and maximize the utilization of its tolerance of overload. The total power consumption can vary a lot along with time, and the gap between CB power and total power is covered by UPS power. In contrast, SGCT-V1 and SGCT-V2 set an upper bound of the total power, while overloading CB and discharging UPS provide the power for sprinting in turn.



(a) SprintCon, average frequency=1 for Interactive & 0.59 for Batch



(b) SGCT-V1, average frequency=0.84 for Interactive & 0.91 for Batch



(c) SGCT-V2, average frequency=0.94 for Interactive & 0.84 for Batch

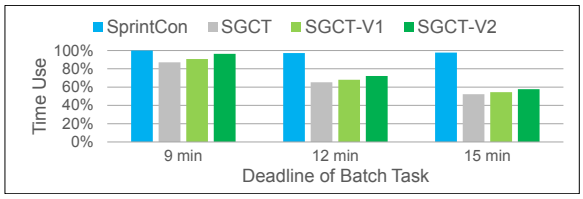
Fig. 7. SprintCon keeps processing interactive workloads at the peak frequency and dynamically adapts the processor cores running batch workloads. Depending on whether overloading CB, the power budget for batch workloads differs a lot, which impacts the frequencies obviously. SGCT-V1 determines the frequency based on processor utilization, and hence the cores running batch workload and utilizing more CPU cycles obtain more chances to sprint. SGCT-V2 gives a higher priority to the interactive workloads, and thus tends to use high frequencies to run them rather than the batch workloads.

significantly throttled, and the average frequency for batch workloads is only 0.59 (normalized value). SGCT-V1 and SGCT-V2 achieve lower average frequencies for interactive workloads (0.84 and 0.94, respectively), but higher average frequencies for batch workloads (0.91 and 0.84, respectively). While SGCT-V2 does allocate more sprinting power to handle interactive workloads than SGCT-V1, it still cannot perform so well as SprintCon, for batch workloads are not so sensitive to processing speed. Considering for SGCT the normalized average frequency for interactive workloads is 0.64, we derive the improvement of computing capacity of SprintCon, which is from (1/0.94 - 1) to (1/0.64 - 1), i.e., from 6% to 56%.

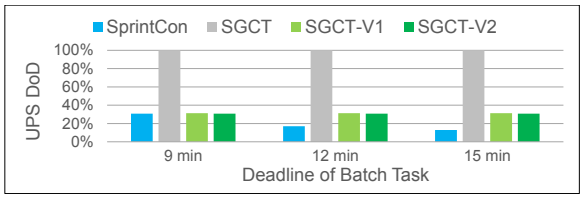
D. Energy Efficiency and Impact of Batch Workload Deadline

As SprintCon generally run batch workloads at lower frequencies than the baseline solutions, it is important to check whether the low frequency and low power consumption can cause misses of deadlines. Here we conduct the simulations with relatively tight deadlines for batch tasks, i.e., 9, 12 and 15 minutes, for which workload deferment is not an option. Figure 8(a) shows that all the tested solutions can have their execution of batch workloads meet the deadlines, but the time used by SprintCon is very close to the deadline, while the baseline solutions still have some redundant time. In general, less time use requires the servers to run the batch workloads with better system performance and consume more power, which not only affects the energy consumption of batch workloads, but can also impact the system performance of the interactive workloads processed simultaneously. Therefore, only SprintCon can efficiently make use of the time before deadlines to save the power consumption of batch workloads, while the baselines run batch workloads unnecessarily fast.

The differences in power behavior and frequency behavior lead to the differences in energy efficiency and cost efficiency. Figure 8(b) shows the total discharge of UPS battery capacity during the 15-minute sprinting process. Although SGCT-V1 and SGCT-V2 do not discharge so much as SGCT, they can still lead to much greater depth of discharge (DoD) than SprintCon, consequently cause much higher battery replacement costs. For example, when the batch workload deadline is 12 minutes, the DoD for SprintCon is only 17%, while the DoD for SGCT-V1 and SGCT-V2 are 31%. According to [32], if LFP batteries are used in the UPS, SprintCon can do such discharge for more than 40,000 battery cycles, but the two baselines can only have less than 10,000 cycles. If the 15-minute sprinting process needs to be conducted 10 time per day, SprintCon do not need to replace the batteries for 10 years, which equals the chemical lifetime of LFP batteries. However, SGCT-V1 and SGCT-V2 need to replace



(a) Comparison on Normalized Time Use



(b) Comparison on UPS Battery DoD

Fig. 8. Batch workloads do not have to be executed immediately but need to be done before the deadlines. (a) shows that all the tested solutions can meet the deadlines, but only SprintCon can efficiently make use of the time before deadlines to save the power consumption of batch workloads. The baselines run batch workloads unnecessarily fast. (b) shows the total usage of UPS capacity. Although SGCT-V1 and SGCT-V2 do not discharge as much as SGCT, they still lead to much greater DoD than SprintCon, and consequently cause much higher battery replacement costs.

the batteries for 3-4 times, and apparently cause much more provisioning costs.

VIII. Conclusion

The computational sprinting methodology has been proposed recently to temporarily improve the performance of data center servers, but existing work has not addressed how to control the sprinting process for safety and efficiency. Those can be significant problems for a data center when computational sprinting is needed for more than a few minutes. In this paper, we propose SprintCon to realize controllable and efficient computational sprinting for data center servers. SprintCon mainly consists of a power load allocator and two different power controllers. The allocator analyzes how to divide the power load to different power sources. The server power controller adapts the CPU cores that process batch workloads, to improve the efficiency in terms of computing, energy and cost. The UPS power controller dynamically adjusts the discharge rate of UPS to satisfy the time-varying power demand of interactive workloads, and ensure power safety. The experiment results show that compared to state-of-the-art solutions, SprintCon can achieve 6-56% better computing performance and up to 87% less demand of stored energy.

References

[1] A. Raghavan, Y. Luo, A. Chandawalla, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin, "Computational sprinting," in *HPCA*, 2012.

[2] S. Fan, S. M. Zahedi, and B. C. Lee, "The computational sprinting game," in *ASPLOS*, 2016.

[3] W. Zheng and X. Wang, "Data center sprinting: Enabling computational sprinting at the data center level," in *ICDCS*, 2015.

[4] A. Raghavan, L. Emurian, L. Shao, M. Papaefthymiou, K. P. Pipe, T. F. Wenisch, and M. M. K. Martin, "Computational sprinting on a hardware/software testbed," in *ASPLOS*, 2013.

[5] A. Rezaei, D. Zhao, M. Daneshtalab, and H. Wu, "Shift sprinting: Fine-grained temperature-aware noc-based mcsoc architecture in dark silicon age," in *DAC*, 2016.

[6] 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.

[7] H. Cai, X. Zhou, Q. Cao, H. Jiang, F. Sheng, X. Qi, J. Yao, C. Xie, L. Xiao, and L. Gu, "Greensprint: Effective computational sprinting in green data centers," in *IPDPS*, 2018.

[8] C. Lefurgy, X. Wang, , and M. Ware, "Power capping: a prelude to power shifting," *Cluster Computing: the Journal of Networks, Software Tools and Applications (Springer), 11(2): 183-195*, 2008.

[9] X. Wang, M. Chen, and X. Fu, "MIMO power control for high-density servers in an enclosure," *IEEE Transactions on Parallel and Distributed Systems, 21(10): 1412-1426*, 2010.

[10] X. Wang, M. Chen, C. Lefurgy, and T. Keller, "SHIP: Scalable hierarchical power control for large-scale data centers," in *PACT*, 2009.

[11] M. Chen, X. Wang, and X. Li, "Coordinating processor and main memory for efficient server power control," in *ICS*, 2011.

[12] Q. Deng, D. Meisner, A. Bhattacharjee, T. Wenisch, and R. Bianchini, "CoScale: Coordinating cpu and memory system dvfs in server systems," in *MICRO*, 2012.

[13] Q. Chen, L. Zheng, M. Guo, and Z. Huang, "EEWA: Energy-efficient workload-aware task scheduling in multi-core architectures," in *IPDPSW*, 2014.

[14] W. Zheng, K. Ma, and X. Wang, "Exploiting thermal energy storage to reduce data center capital and operating expenses," in *HPCA*, 2014.

[15] C. Li, Z. Wang, X. Hou, H. Chen, X. Liang, and M. Guo, "Power attack defense: Securing battery-backed data centers," in *ISCA*, 2016.

[16] X. Hou, L. Hao, C. Li, Q. Chen, W. Zheng, and M. Guo, "Power grab in aggressively provisioned data centers: What is the risk and what can be done about it," in *ICCD*, 2018.

[17] Rockwell Automation. [Online]. Available: http://literature.rockwellautomation.com

[18] X. Fu, X. Wang, and C. Lefurgy, "How much power oversubscription is safe and allowed in data centers?" in *ICAC*, 2011.

[19] 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.

[20] H.-P. D. Company, "HP power capping and HP dynamic power capping for ProLiant servers," 2011, http://h20565.www2.hp.com/hpsc/doc/public/display?docId=emr_na-c01549455.

[21] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *HPCA*, 2008.

[22] Z. Liu, Y. Chen, C. Bash, A. Wierman, D. Gmach, Z. Wang, M. Marwah, and C. Hyser, "Renewable and cooling aware workload management for sustainable data centers," in *SIGMETRICS*, 2012.

[23] H. Yang, A. Breslow, J. Mars, and L. Tang, "Bubble-flux: Precise online qos management for increased utilization in warehouse scale computers," in *ISCA*, 2013.

[24] W. Zheng, K. Ma, and X. Wang, "Hybrid energy storage with supercapacitor for cost-efficient data center power shaving and capping," *IEEE Transactions on Parallel and Distributed Systems*, 2017.

[25] 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," in *MICRO*, 2006.

[26] R. Teodorescu and J. Torrellas, "Variation-aware application scheduling and power management for chip multiprocessors," in *ISCA*, 2008.

[27] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *ISCA*, 2009.

[28] K. Ma, X. Wang, and Y. Wang, "DPPC: Dynamic power partitioning and control for improved chip multiprocessor performance," *IEEE Transactions on Computers, 63(7): 1736-1750*, 2014.

[29] T. Horvath and K. Skadron, "Multi-mode energy management for multi-tier server clusters," in *PACT*, 2008.

[30] J. M. Maciejowski, "Predictive control with constraints," *Prentice Hall*, 2002.

[31] G. Urdaneta, G. Pierre, and M. Steen, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, 2009.

[32] V. Kontorinisy, L. Zhangy, B. Aksanliy, J. Sampsony, H. Homayouny, E. Pettisz, M. Tullseny, and T. Rosing, "Managing distributed ups energy for effective power capping in data centers," in *ISCA*, 2012.