Data Center Sprinting: Enabling Computational Sprinting at the Data Center Level

Wenli Zheng and Xiaorui Wang
Department of Electrical and Computer Engineering
The Ohio State University, Columbus, OH, USA
Email: {zheng.691, wang.3596}@osu.edu

***Abstract*—Microprocessors may need to keep most of their cores off in the era of dark silicon due to thermal constraints. Recent studies have proposed Computational Sprinting, which allows a chip to temporarily exceed its power and thermal limits by turning on all its cores for a short time period, such that its computing performance is boosted for bursty computation demands. However, conducting sprinting in a data center faces new challenges due to power and thermal constraints at the data center level, which are exacerbated by recently proposed power infrastructure under-provisioning and reliance on renewable energy, as well as the increasing server density.**

In this paper, we propose Data Center Sprinting, a methodology that enables a data center to temporarily boost its computing performance by turning on more cores in the era of dark silicon, in order to handle occasional workload bursts. We demonstrate the feasibility of this approach by analyzing the tripping characteristics of data center circuit breakers and the discharging characteristics of energy storage devices, in order to realize safe sprinting without causing undesired server overheating or shutdown. We evaluate a prototype of Data Center Sprinting on a hardware testbed and in datacenter-level simulations. The experimental results show that our solution can improve the average computing performance of a data center by a factor of 1.62 to 2.45 for 5 to 30 minutes.

***Keywords*-data center; computational sprinting; energy storage; circuit breaker; power management.**

I. Introduction

As transistor density continues to increase, a microprocessor chip is expected to soon have more transistors than can be sustainably powered up due to thermal constraints. As a result, some of the processor cores must remain off most of the time. This phenomenon is commonly known as dark silicon or the utilization wall [4], [7]. For example, it has been predicted that over 50% of the computing resources may have to stay inactive in 2024 [7]. While the cores of a chip cannot be all turned on in a sustainable way, recent studies have proposed a concept called computational sprinting [32], [31], which allows a chip to temporarily exceed its power and thermal limits by turning on all its cores for a short time period, after which the chip immediately returns to normal operation (with most cores shut down) to cool down. Computational Sprinting can effectively boost the computing performance of a computer system to handle occasional computation demand bursts (high but short-lasting) that are becoming more common to both mobile devices and servers.

Figure 1 shows the trace of aggregated traffic rates of 1,500 servers in a Microsoft data center [17], which is directly related to the demand of computing resources. We can see that the demands are indeed bursty even for such a data center with mostly throughput-oriented workloads. For data centers with more interactive workloads (e.g., search, forum, news), workload bursts can be less frequent but higher in a variety of circumstances (e.g., breaking news). In addition to bursty workloads, another similarity with chip-level sprinting is that data centers also need to have some of their cores shut down most of the time in the foreseeable future for several reasons: 1) increasing server density, which causes more stringent thermal constraints, 2) the recent trend of under-provisioning the power infrastructure [2], [8], [12], [13], [42], which leads to less provisioned power capacity, and 3) increasing reliance on the intermittent renewable power supplies [23], [21]. More importantly, in the era of dark silicon, it is likely that most microprocessors available in the market already have inactive cores provisioned due to chip-level power and thermal constraints [7]. Therefore, a future data center may need to keep many of its cores off for considerably long periods of time, and can conduct Computational Sprinting to dynamically turn on those inactive cores for short durations when computation demand bursts arrive.

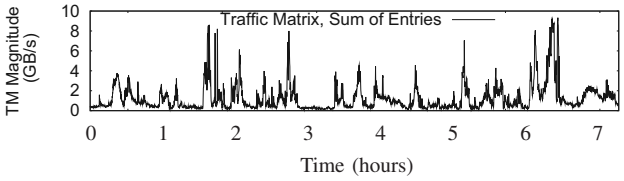


Fig. 1. A traffic trace of 1,500 servers in a Microsoft data center, which is directly related to the demand of computing resources [17]. The Y-axis indicates the aggregated traffic rate.

Compared to sprinting for low-end mobile devices [32], [31] that are sensitive to costs, data centers are more willing to provision more cores than can be powered up, just to temporarily service any computation bursts. The key reason is that rejecting user requests due to power capacity may cause data centers to lose revenue or even customers in the long run. Unfortunately, today's data centers are already approaching the peak capacity of their power infrastructures, but upgrades can be extremely expensive (ranging in the hundreds of millions USD) [43]. Therefore, sprinting can be profitable for data centers. As shown in Section V-D, the revenue earned by sprinting can significantly exceed the costs of provisioning those extra cores that are normally inactive.

Sprinting at the data center level faces two major new challenges. First, the power limits are not just the Thermal

This work was supported, in part, by NSF under grants CCF-1143605, CCF-1331712, and CNS-1143607 (NSF CAREER Award).

Design Power (TDP) at the chip level, where chip temperature is the main concern and adequate power is always supplied during a sprinting process. In a data center, simply conducting chip-level sprinting without appropriate control can overload various components in the power infrastructure, like the power distribution units (PDUs) and the on-site power substation, tripping their circuit breakers (CBs) and leading to disastrous server shutdowns. Second, the thermal constraints are not only the chip-level ones that can already be handled by a package of phase-change materials (PCM) attached to the chip [32], [31]. In a data center, the cooling system needs to use more power to effectively remove the extra heat dissipated from chip-level sprinting. However, in the meantime, the IT equipment (servers, networks) also consumes more power than usual, while the total power capacity of the entire data center is limited. Hence, how to enhance cooling for sprinting with even less cooling power is challenging.

In this paper, we propose *Data Center Sprinting*, a three-phase methodology that enables safe computational sprinting at the data center level, to temporarily boost the performance during occasional workload bursts. Data Center Sprinting significantly differs from previous work on power capping, because it provides more power to servers than normally allowed, instead of throttling their power when they need it the most. Data Center Sprinting is also significantly different from existing work on data center power provisioning [12], [13], [41], [46] that tries to reduce the data center capital or operating expenses, because sprinting is designed to temporarily boost the computational capacity for better application performance within a given power infrastructure capacity.

To address the challenge of violating the power limits at the PDU and substation levels, we explore the tripping characteristics of data center CBs and find they (as well as other power infrastructure components protected by the CBs) are actually designed to sustain a certain amount of power overload. According to the UL489 standard, violating the rated power limit for a short duration is allowed, as long as the magnitude and duration of power overload is appropriately controlled. The power loads of the data centers in the US are conservatively configured based on the National Electric Code (NEC) [29] to have more than sufficient headroom. Such control has already been used in practice to handle unstable power swing, but not yet for temporary performance boost. Consequently, with real-time monitoring and control, such headroom enables instantaneous sprinting and allows time for a data center to get additional power from energy storage devices (ESDs), such as uninterruptible power supply (UPS) batteries, to support sprinting. In addition, we also explore auxiliary ESDs commonly equipped in data centers, such as thermal energy storage (TES) tanks and compressed air, to extend the sprinting duration. In this paper, we use TES tanks (which contain cold materials such as cold water or ice) as an example because they are already installed in many data centers [25] and can provide additional cooling for sprinting.

Specifically, Data Center Sprinting is designed as a three-phase methodology: The first phase relies on the CB tolerance of power overload for instantaneous sprinting before ESDs are activated; the second phase uses UPS to supply additional power when extra cooling is not yet needed; the third phase activates TES to enhance cooling and reduce chiller power. To maximize the average computing performance during a workload burst, we design four strategies to dynamically determine how many cores should be activated. Specifically, this paper makes the following major contributions:

- We propose Data Center Sprinting, a methodology that enables a data center to temporarily boost its computing performance by turning on more cores in the era of dark silicon, in order to handle occasional short-lasting but high workload bursts. We quantitatively show that sprinting can be profitable for data centers.
- We demonstrate the feasibility of this approach by analyzing data center CBs and two commonly used ESDs: UPS and TES. We show that Data Center Sprinting improves not only the instantaneous performance but also the power safety of a data center, because uncontrolled chip-level sprinting causes undesired shutdowns (Section VII-A).
- We design four sprinting strategies to dynamically determine how many cores should be activated for sprinting based on time-varying workload demands, to optimize the computing performance.
- We evaluate a prototype of Data Center Sprinting on a hardware testbed and in datacenter-level simulations. The experimental results show that our solution can improve the average computing performance of a data center by a factor of 1.62 to 2.45 for 5 to 30 minutes.

The rest of this paper is organized as follows. Section II reviews the related work. Section III introduces background knowledge. Sections IV and V present an overview and detailed designs of our methodology. Section VI describes the evaluation methodology while Section VII analyzes the experiment results. Section VIII concludes the paper.

II. Related Work

The concept of Computational Sprinting has been first proposed by Raghavan et al. [32]. Based on the observation that microprocessors will have most of their cores turned off due to thermal constraints, they propose to temporarily turn on all the cores to boost the performance of bursty workloads. In order to address chip overheating caused by sprinting, they have designed a package containing phase-change materials for enhanced chip-level cooling. Later in [31] Raghavan et al. have successfully implemented a prototype of Computational Sprinting on a hardware/software testbed. In this paper, we assume that computational sprinting has already been applied to the processor chips in a data center and propose to address the new challenges at a higher level: the data center level, in order to avoid undesired server overheating or shutdown.

Power infrastructure under-provisioning has recently received a lot of attention. The studies from Fan et al. [8] and Barroso et al. [2], based on the workload analysis in Google data centers, have shown that the power infrastructures of today's data center are over-provisioned to prepare for the worst-case workloads that rarely occur. They have suggested that the capital expenses of a data center can be

significantly reduced if the power infrastructure can be over-subscribed. A lot of recent research has proposed a variety of ways, mostly by utilizing the UPS batteries equipped in data centers, to over-subscribe the data center power infrastructure as effective implementations of under-provisioning [12], [13], [18], [41]. In this paper, we assume that a future data center has already implemented under-provisioning and discuss how to temporarily boost the data center performance, given the limited capacity of an under-provisioned power infrastructure. Fu et al. [10] have analyzed the capacity of circuit breakers to tolerate power overloads. In contrast, we integrate circuit breakers with UPS and TES devices for a three-phase solution that enables computational sprinting at the data center level.

Another group of recent studies that is closely related to our work is power capping, which aims to optimize the computing performance of a computer system within a given power cap. For example, power capping algorithms have been proposed at the chip level [15], [24], the server level [20], the cluster level [34], [44], and the data center level [33], [43]. Almost all the aforementioned power capping work relies on dynamic voltage and frequency scaling (DVFS) as a main knob to ensure that the power consumption never exceeds the given cap. In contrast, we propose to temporarily violate the power limits by turning on more cores than allowed to boost computing performance for short time durations. Therefore, our solution can result in much better performance for busty workloads.

III. Background

In this section we briefly introduce the background information of several technologies adopted by modern data centers, which are exploited by Data Center Sprinting.

A. Circuit Breaker

In data centers, the power infrastructures (e.g., the on-site power substation and the PDUs) are protected by their circuit breakers (CB). Intuitively, the rated current of a CB is usually considered as the maximum sustainable current that is safe to the protected power delivery component. Hence to avoid power outage, the common practice is to constrain the power consumption below the rated power capacity, which is however too conservative in fact. According to [39], [10], many components (e.g., transformer in the substation, generator, UPS and various cables) in the power infrastructure can tolerate some power overload for a certain period mainly because of unstable swing. CBs are also designed to allow some overload to avoid unnecessary trip-up. The relationship between the magnitude of a power overload and the trip time of a Bulletin 1489-A Circuit Breaker [10], [35] is shown in Figure 2.

B. UPS Battery

Uninterruptible power supply (UPS) devices are widely equipped in data centers to temporarily supply power when the main power source (e.g., the utility power grid) suddenly fails and before the diesel generator starts to work. The used battery capacity can be recharged later when the power demand is low. While the startup of diesel generator usually takes tens of seconds, the UPS can usually keep working for several minutes. Hence the over-provisioned capacity can be exploited

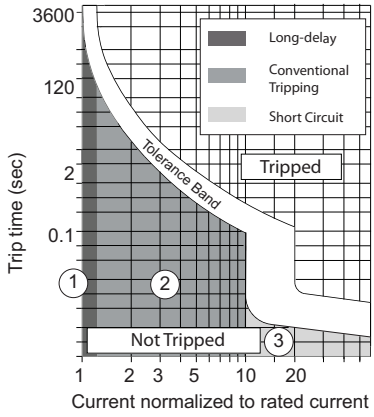
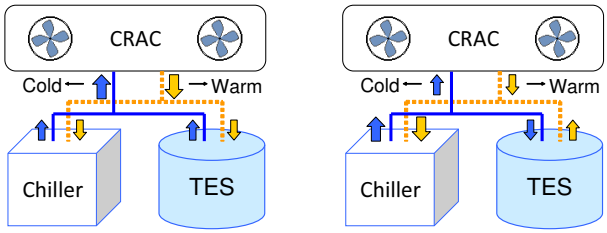


Fig. 2. The trip curve of a typical circuit breaker [35], [10]. The trip time decreases when the overload current increases.



(a) Discharge (b) Recharge

Fig. 3. The directions of coolant flows when discharging / recharging a TES tank that stores cold coolant.

for other purpose, based on the previous studies [12], [13], [18], [41], which also show infrequent uses of batteries do not shorten their lifetime to be less than their required lifetime (e.g., 4 years for LA and 8 years for LFP).

C. TES Tank

Many existing data centers rely on chiller-based computer room air conditioning (CRAC) systems for cooling; even some data centers applying free cooling technologies (e.g., using cold outside air for cooling) still employ chillers as backup since the free cooling scheme may not work all the time (e.g., the outside air might be too hot during the daytime in summer). Thermal energy storage (TES) tanks, which store cold materials like cold coolant or phase-change materials (PCMs) that can be used for cooling, are often equipped together with chillers as a backup. Since TES is much less costly than UPS, it is more likely for data centers to over-provision the TES as they already have excessive UPS capacity.

The working principle of a TES tank storing cold coolant is described in Figure 3.When discharging the TES, the CRAC unit can get more cold coolant than provided by the chiller. Hence the cooling effect inside the data center is enhanced, or the chiller can be tuned to consume less power without affecting the cooling of data center. When recharging the TES, the chiller produces more cold coolant than consumed by the CRAC, and the remaining amount is stored in the TES tank.

IV. Overview of Data Center Sprinting

We first have an overview of our design, considering a data center hosting multi-core servers, but normally only part of

the cores are active due to the dark silicon phenomenon. To handle occasional workload bursts, more cores are turned on to boost the processing capability, requiring for additional power and cooling. Similar to the studies on chip-level sprinting [32], [31], we focus on the compute-intensive workloads in this paper, which can immediately benefit from the increase of active cores. Sprinting is only used to handle delay-sensitive workloads, since the delay-insensitive workloads can be postponed. When more cores are activated for Data Center Sprinting, the prerequisite is that the chip-level sprinting is already safely enabled. If the chip-level sprinting can be no longer sustained, we also finish Data Center Sprinting.

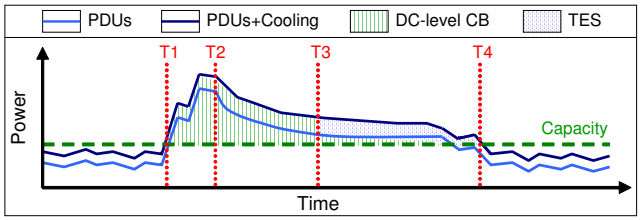
A. Sprinting Degree and Duration

There are two important aspects for a sprinting process, *sprinting degree* and *sprinting duration*. Sprinting degree means the ratio of the number of active cores during sprinting over the number of active cores in the normal situations. In this paper, we propose four strategies to dynamically determine the upper bound of sprinting degree, while the real sprinting degree can be lower if the workload does not fully utilize all the active cores. Sprinting duration means how long the sprinting process can be sustained. A sprinting process can finish for two reasons: 1) the workload burst finishes; 2) the additional power or cooling can no longer be provided. The latter can be due to lack of stored energy, or some special cases that occur during the sprinting process, such as unexpected power spikes in the utility power supply. When these issues lead to higher CB overload, which can be detected with real-time power measurement, we immediately lower the sprinting degree or end sprinting to ensure the power safety of the data center. In general, a higher sprinting degree leads to higher power consumption and hence a shorter sprinting duration.

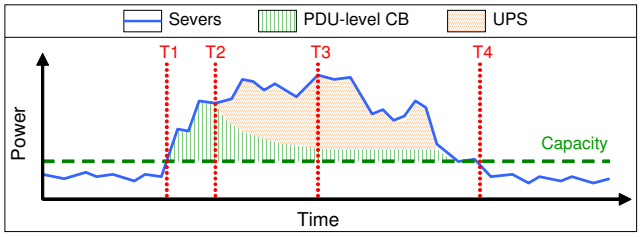
B. Additional Power and Cooling

Data Center Sprinting exploits CB tolerance and two types of energy storage devices (ESDs), UPS and TES, to provide the additional power and cooling. The two ESDs are already equipped by many data centers, and thus we do not need extra ESDs. In this paper, we assume UPS batteries are distributed at the server level, which is considered as an efficient way of UPS deployment [18]. The distributed UPS batteries can be coordinated to set a desired number of servers to be powered by their batteries, as discussed in [18], to adjust the power delivered through the PDUs. Since a UPS battery (e.g., LFP battery) can be fully discharged for 10 times per month without its lifetime being affected, according to [18], we can apply it to handle occasional workload bursts without additional battery cost. We also assume a CRAC system is used for cooling, which primarily relies on chillers to produce the coolant. During a sprinting process, we use TES to enhance cooling and even reduce the chiller power. For PDUs, CRAC and TES, since Data Center Sprinting is designed to handle occasional bursts with relatively short durations, the impacts of sprinting on their lifetimes can be considered to be negligible.

Figure 4 generally illustrates how the additional power is provided at the data center level and the PDU level. The



(a) At the data center level



(b) At the PDU level

Fig. 4. The general illustration of the three-phase methodology. The CB is overloaded and the TES (a) or UPS (b) is discharged to supply the additional power, when the power demand of the PDUs & cooling system (a) or the servers (b) exceeds the power capacity. Phase 1: T1 to T2; Phase 2: T2 to T3; Phase 3: T3 to T4.

curves $PDUs$ and $PDUs+Cooling$ in Figure 4(a) represent the power demands of the PDUs and the entire data center, respectively. The curve $Servers$ in Figure 4(b) represent the power demand of the servers in a PDU group. From T1 to T4 in Figures 4(a) and (b), Data Center Sprinting can be divided into three phases. When the workload burst comes, the first phase (from T1 to T2) starts, during which we rely on overloading the CBs at both the two levels to supply extra power. As we gradually decrease the upper bound of CB overload, the second phase (from T2 to T3) starts when overloading the CBs cannot provide adequate power, to discharge the UPS to supply part of the serve power. In the third phase (from T3 to T4), which starts before the data center is overheated, the TES enhances the cooling and even saves some chiller power to reduce the DC-level CB overload. The specific time to start each phase depends on sprinting degree, CB headroom, cooling condition and some other configurations. The detailed operations of Data Center Sprinting is to be discussed in Section V.

V. Design

In this section, we present our four strategies that dynamically determine the upper bound of sprinting degree, and how the three knobs are used to provide additional power and cooling. For some data centers without TES, we are unable to save chiller power but can still enable sprinting (though the duration is shorter), since the devices and air in data centers have some thermal capacitance to mitigate temperature increase [45]. We also analyze the cost of provisioning normally inactive cores and the revenue of sprinting, to show Data Center Sprinting can be profitable for data centers.

A. Strategies to Determine Sprinting Degree

The first step in each operation period is to determine the sprinting degree, i.e., how many additional cores to activate.

Based on previous studies on workload prediction [5], [38], the simplest solution is to activate just enough cores according to the workload demand. When we can no longer sustain the power or cooling, we finish sprinting by putting the additional active cores back to be inactive. We call it the *Greedy* strategy. If the workload burst requires more cores than the data center has, or continues for a longer time than the sprinting duration, we have to deny part of the requests with admission control like [3], which is the last resort.

However, sometimes constraining the sprinting degree to extend the sprinting duration can achieve better overall performance. For example, if a burst lasts for 10 minutes but the Greedy strategy can sustain for only 6 minutes, we have to drop about 40% of the requests. If we constrain the sprinting degree to handle 80% of the workload demand, the sprinting duration may be extended to 9 minutes, and we only have to drop 28% of the excess requests. This can be explained by our tests about running SPEC jbb2005 benchmarks on a quad-core i5 processor, whose results show that the per-core throughput decreases when the number of cores increases, i.e., a lower sprinting degree can have a higher power efficiency.

Therefore, we design three other strategies to determine the upper bound of sprinting degree when adequate power and cooling are supplied (otherwise the available power and cooling become the bottlenecks to constrain the sprinting degree). The *Oracle* strategy finds the optimal upper bound by exhaustive search, with the assumption that the burst degree and burst duration can be perfectly predicted. This strategy is actually impractical but provides a reference for the other strategies to compare with. We can also use the Oracle strategy to make an upper bound table, listing the optimal upper bounds for different burst durations and maximum burst degree.

The *Prediction* strategy works with a *predicted burst duration* (BDu_p). It dynamically calculates the average sprinting degree (SDe_{avg}) from the beginning of the burst to the current moment, and derives the equivalent burst duration BDu_e:

$$BDu_e(t) = BDu_p \times (SDe_{max}/SDe_{avg}(t)) \quad (1)$$

where SDe is the real sprinting degree and SDe_{max} is its maximum allowed value. Based on $BDu_e(t)$, this strategy selects the optimal upper bound of sprinting degree ($SDe_{opt}(t)$) from the upper bound table, and $SDe(t) \leq SDe_{opt}(t)$.

The *Heuristic* strategy works with an *estimated best average sprinting degree* (SDe_p), based on the prediction of bursts as well. It increases this value by a flexibility factor $K\%$ to derive the initial upper bound of sprinting degree SDe_{ini}, where $K\%$ is a user-defined parameter to specify how strict the upper bound should be. Then this strategy dynamically adjusts the upper bound $SDe_u(t)$ as:

$$SDe_u(t) = SDe_{ini} \times (RE(t)/RT(t)) \quad (2)$$

where RE indicates the remaining energy and RT indicates the remaining time. They are calculated based on the remaining energy budget (EB), total energy budget (EB_{tot}, which is sum of stored energy and the additional energy delivered by overloading the CBs) and predicted sprinting duration ($SDu_p = EB_{tot}/SDe_p$):

$$RE(t) = EB(t)/EB_{tot},\ RT(t) = (SDu_p - t)/SDu_p \quad (3)$$

In the above calculation, we regard the sprinting degree as a continuous variable, while in fact it is discrete with a fine granularity (each core can be individually powered on or off). Those strategies can help us to find how constrained sprinting degree can improve the performance in different situations. To further optimize the sprinting degree, we can develop more sophisticated strategies by integrating some recently proposed solutions for burst prediction (e.g., [19], [36]) and formulate optimization problems to minimize the performance degradation, which is our future work.

B. Phases 1 and 2: CB and UPS Involved

The first phase of Data Center Sprinting (from T1 to T2 in Figures 4(a) and (b)) starts immediately after additional cores are activated, and ends when UPS joins in power supply. To prevent tripping the CBs with too much power overload, we limit the total power of all the downstream branches under the upper bound. For example, if the power overload of a parent CB has already reached its upper bound, then a power increase on any of its child CBs demands a power decrease on some other child CBs, in order to keep their sum unchanged. Therefore, we never trip a CB at the substation level by overloading the CBs at the PDU level.

While the UPS can be started within several milliseconds [41], we may select not to do so, in order to save battery energy, especially when the burst duration is relatively short. However, for longer bursts, discharging UPS can effectively reduce the CB overload, extend the time before trip-up, and improve the overall performance. In our design, we initially set a high upper bound of CB overload to allow the maximum sprinting degree, assuming the burst duration might be short. Then we gradually decrease the upper bound as time goes, since the burst could be relatively long. Specifically, we dynamically calculate the remaining time before the CB trips if the current overload continues. If the remaining time is less than 1 minute, we decrease the upper bound of CB overload until the remaining time equals to 1 minute. Note here the 1 minute is a user-defined parameter, which can be adjusted to change how aggressively we want to overload the CBs.

Therefore, after some time the sprinting degree that can be sustained by overloading the CBs would be less than the upper bound of sprinting degree. At this moment, we step into the second phase of Data Center Sprinting (from T2 to T3 in Figure 4(a) and (b)), and discharge power from the UPS. The UPS output power equals to the difference between the real power demand and the power delivered through the CB, which gradually increases as the upper bound of CB overload decreases, as shown in Figure 4(b).

C. Phase 3: TES Involved

To determine when to enter the third phase and activate the TES to avoid overheating, the operator needs to consider how fast the data center temperature would increase due to sprinting. Since the thermodynamics of a data center depends on its specific rack layout and AC locations, Computational Fluid Dynamics (CFD, a widely adopted method to analyze the fluid dynamics, such as the complex thermal conditions in

data centers) can be used to estimate the temperature changes. Schneider Electric's Data Center Science Center has studied how the air temperature in a data center rises if the chiller stops to work, based on detailed CFD simulations [22]. After the failure of chiller, there exists a gap between the heat generation rate (determined by the power consumption of servers) and the heat absorption rate (falling to zero). Their results show that the temperature threshold will never be achieved if the chiller is resumed (i.e., filling up the gap) at the 5th minute.

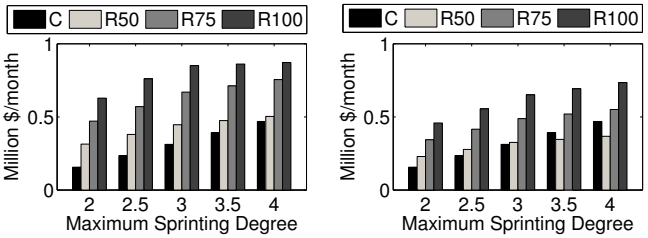
For simplicity, we assume the same thermodynamics of data center in this paper to apply their CFD results, though in practice each data center needs to do their own CFD simulation. Since we do not raise the chiller power in the first and second phases, if the additional server power is equal to the normal peak server power, the gap between the heat generation and absorption rates is close to the gap in the above CFD scenario. In this case, we can also activate TES to enhance cooling at the 5th minute. In general, we calculate the time to activate TES as (5 minute $\times$ normal peak server power / maximum additional server power), assuming the speed of temperature increase is proportional to the additional server power. Here the maximum value is a conservative setting, and the actual temperature increase can be slower.

When the TES kicks in, we step into the third phase of Data Center Sprinting (from T3 to T4 in Figures 4(a) and (b)). We exploit TES not only to enhance cooling, but also reduce the chiller power to decrease the overload of DC-level CBs, as shown in Figure 4(a). According to [16], up to 2/3 of the cooling power can be saved by using TES to replace the chiller, while the rest 1/3 is consumed by the pumps, valves and CRAC fans. If the TES capacity is used up, we need to terminate the sprinting process by resuming the chiller and decreasing the number of active cores to the normal level, in order to avoid overheating.

D. Cost and Revenue

Data Center Sprinting exploits the ESDs existing in many data centers and avoids additional expenses on ESDs. Since workload bursts come occasionally (e.g., less than 10 times per month), no extra cost of UPS is needed. The major potential cost is to provision more cores than can be sustainably powered up, which are well expected for future data centers in the dark silicon era. Compared to low-end mobile devices [32], [31] that are sensitive to costs, data centers are more willing to provision those normally inactive cores, in order to process temporary workload bursts. The key reason is that rejecting user requests due to power capacity may cause data centers to lose revenue or even customers in the long run. Here we quantitatively compare the cost of additional cores and the revenue of sprinting.

To analyze cost, we assume there are 10 normally active cores on the chip of each server, like the Intel Xeon 10-core processors used by Amazon EC2 servers [1]. Provisioning each additional core costs about $40 [37]. If this cost is amortized over 4 years (48 months), the monthly per-server cost of the additional cores is $\$40 \times (10 \times N - 10)/48 = \$8.3(N-1)$, where N is the maximum sprinting degree, i.e., the total number of cores normalized to the number of normally active cores ($N = 1$ means no additional cores). We estimate the number of servers hosted by a data center based on [40], [28], in which a relatively small data center may have 12,500 servers. For a relatively large data center, we assume it has 25,000 servers on average, considering Google is reported to have 36 data centers [26] and 900,000 servers [27]. We calculate the average number of servers hosted by a data center as $(25,000 + 12,500)/2 = 18,750$, and the monthly per-datacenter cost as $\$8.3(N-1) \times 18,750 = \$156,250(N-1)$.



(a) 4X typical # of users ($U_t = 4U_0$) (b) 6X typical # of users ($U_t = 6U_0$)

Fig. 5. Average cost and revenue of Data Center Sprinting with three 5-min workload bursts per month. C indicates the cost. Rxx indicates the revenues with different burst magnitudes that utilize $xx\%$ of the additional cores.

We now analyze the potential revenue that can be earned by sprinting. According to a survey from the Ponemon Institute [40], an average-scale data center (we assume it hosts 18,750 servers as calculated above) can lose $7,900 per minute for being unable to provide service. Similarly, when a data center has to deny a portion of the requests (e.g., due to inadequate active cores), it means the service is temporarily unavailable to them. Thus we assume the revenue loss is proportional to the amount of dropped requests.We calculate the monthly revenue of sprinting for handling extra requests as $\$7,900 \times L \times (M-1) \times K$, where L is the burst duration in minutes, K is the number of bursts, and M is the average magnitude of workload bursts normalized to the maximum workload that can be handled without sprinting (sprinting is not needed for $M \leq 1$).

In addition, data centers without sprinting can even have more revenue loss due to losing customers, which is a long-term impact on their business. A test conducted by Google [9] shows that they can permanently lose 0.2% users when they have the response time increased by only 0.4 seconds. We calculate the monthly revenue loss due to losing 0.2% users as $\$7,900 \times 43,200 \times 0.2\% = \$682,560$, as there are 43,200 minutes in a month. The average per-user amount is $\$682,560/U_t$, where U_t is the total number of users of the data center. We estimate the number of users whose requests are dropped during some bursts without sprinting as $min[U_0(M-1)K, U_t]$, where U_0 is the maximum number of users whose requests can be processed by the data center simultaneously without sprinting. Thus the monthly revenue due to retaining customers is $(\$682,560/U_t) \times min[U_0(M-1)K, U_t]$. This estimation is conservative because workload bursts can lead to much longer delays or even dropping the requests.

Therefore, the total revenue of sprinting includes the revenue of handling more requests during workload bursts, and

the revenue of retaining customers. For example, a data center has the workload in Figure 1 and it repeats for a month. If the rated power capacity supports processing 4 GB/s of the workload traffic at most, the monthly revenue of sprinting with $N = 4$ and $U_t = 4U_0$ is about \$19 Million, equal to 5.6% of the total monthly revenue without sprinting, while the monthly cost of additional cores is only \$0.47 M. To stress test workloads with much fewer bursts, we configure $L = 5min$, $K = 3$, $U_t = 4U_0$, and show the costs and revenues in Figure 5(a). Different magnitudes of workload bursts are considered, which utilize 50%, 75% or 100% of the additional cores. If the bursts are relatively low (e.g., 50%), the profit becomes less with more additional cores, which support a greater value of N but are not sufficiently utilized. If the bursts are high and sufficiently utilize the additional cores, sprinting can make a monthly profit of more than \$0.4 M. The revenue of sprinting is much lower than \$19 M because we have only 3 bursts per month in this case, while the workload in Figure 1 has 200 bursts in a month that discharge 26% of the UPS capacity each time on average, which has no impact on UPS lifetime according to [18]. When the data center has more users, e.g., $U_t = 6U_0$ in Figure 5(b), the extra revenue due to reducing customer loss may become less, because the impact of handling the same bursts on all the users becomes less. However, having more users means that the burst degree is likely to be higher, and thus sprinting can still be profitable.

VI. Evaluation Methodology

In this section, we introduce our evaluation methodology.

A. Simulation Setup

In the simulation, we configure each server as a 48-core Single-chip Cloud Computer proposed by Intel [14], whose processor consumes 125 W when fully utilized, considering future servers are likely to have many-core chips. Each fully utilized core consumes 2.5 W, and the chip consumes 5 W when all cores are inactive. The non-CPU power (consumed by other server components) is constantly 20 W, which may have some fluctuation in practice. The relatively small non-CPU power is a conservative configuration. If it is larger, fewer servers can be hosted within the same power capacity, and the power for sprinting would be lower and lead to longer sprinting duration. In this data center, each server normally runs 12 cores, and the peak normal power (i.e., the peak power without sprinting) is $20W + 5W + 12 \times 2.5W = 55W$.

We model a data center whose peak normal power consumption of all the servers can be 10 MW (the same as [18]). Thus in the whole data center about 180,000 servers can be hosted. The cooling power is calculated based on the power consumption of servers and the Power Usage Effectiveness (PUE). We assume the PUE is 1.53 according to [30] in default, considering only server power and cooling power, and test different PUE values. Therefore, in default the rated power limit of DC-level power infrastructure should be $15.1MW \times (1 + 25\%) = 19MW$, if it is provisioned based on the peak normal power, and the headroom is 25% as mentioned in Section III-A. However, due to power under-provisioning, the real headroom can be less than 25% of the peak normal power. We set the default headroom to be 10% of the peak normal power, and test it from 0 to 20% in the simulation. Inside the data center, each PDU delivers power to 200 servers as in [18]. And hence the rated power limit of a PDU-level CB is $55W \times 200 \times (1 + 25\%) = 13.75kW$.

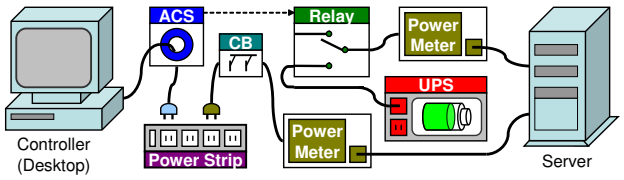


Fig. 6. Hardware experiment Testbed. The ACS controls the relay based on the controller output. One power socket of the server can draw power from the UPS or be cut off, depending on whether the relay connects the circuit. Another power socket is fixedly connected to the power strip through the CB.

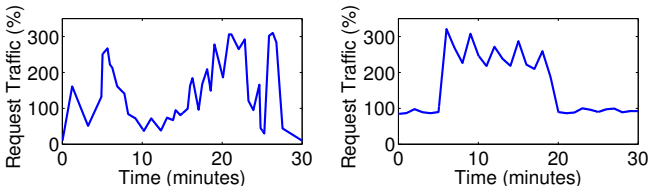
We use the trip time curve of Bulletin 1489 [35] circuit breaker, and assume the UPS batteries are distributed at the server level as in [18]. The default battery capacity is 0.5 Ah, which can sustain the peak normal power of a server (i.e., 55 W) for about 6 minutes, since a UPS runtime between 1 to 10 minutes is common [18], [12]. The TES tank is able to take over the cooling load for 12 minutes when the servers consume the peak normal power, as suggested by [11].

B. Hardware Experiment Setup

To evaluate our design in a practical environment, we build a prototype hardware testbed. We make a server able to draw power from two sources: a power strip and a UPS device. We use a server that has two power sockets. The first one is connected to the power strip through a CB, while the second one is connected to a relay. The relay is controlled by an AC switch: when the AC switch outputs a high voltage, the relay connects the second socket of the server to the UPS; when the AC switch outputs a low voltage, the relay cuts off this connection. When the second power socket is connected to the UPS, the two power sources supply power to the server together (the two power demands are approximately equal); otherwise the server only uses the first power socket, and the power strip supplies all the server power. Since the switching operation can be completed in less than 10 milliseconds, and the server can tolerate more than 30 milliseconds of power interruption, the switching of power source should not affect the server operation. The AC switch is controlled by another computer (called the *controller* here). The power demands of the server are measured by two Watts Up power meters.

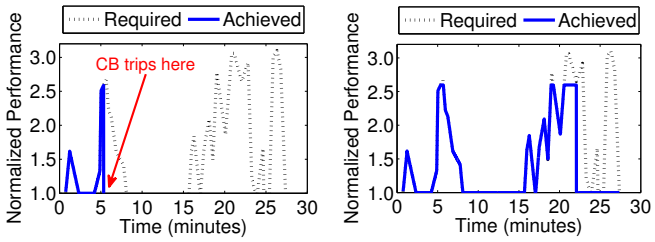
C. Workload

The workloads used in the experiments are generated based on the real trace files from Microsoft [17] and Yahoo! [6] data centers. To obtain the *MS* trace (Figure 7(a)), we first cut out a 30-minute piece from the traffic trace in Figure 1, containing the consecutive bursts from the 71,188th second to the 72,987th second. While the peak traffic rate can be more than 9 GB/s, we assume the peak computing performance without sprinting can handle 3 GB/s, which is adequate for most of the time. Then we normalize the traffic rate to 3 GB/s and get a workload trace in terms of percentage.



(a) *MS* Trace (b) *Yahoo* Trace

Fig. 7. Workload traces for experiments. The *Yahoo* trace can have different burst degrees and durations. The y-axis indicates the workload demand normalized to the normal peak demand.



(a) Uncontrolled chip-level sprinting (b) DC Sprinting with Greedy

Fig. 8. The required and achieved performance with the *MS* workload trace under the default settings. Uncontrolled chip-level sprinting trips the CB after 5 min 20 sec, resulting the shutdown of the data center.

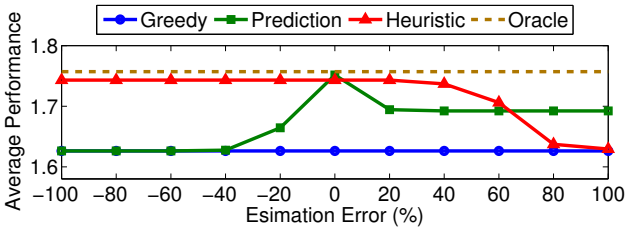


Fig. 9. The average performance of the four strategies with the *MS* workload trace under the default settings. The Prediction and Heuristic strategies are affected by the estimation error, while Greedy and Oracle are not.

The trace file of Yahoo! contains the trace of each server (70 servers in total), recording the user request rates. We aggregate the 70 workload traces together and cut out a 30-minute piece from it, which contains the highest request rate. Different from the *MS* trace, the request rate of the aggregated Yahoo! trace does not change so severely. To test the effectiveness of DC Sprinting for different burst degrees and durations, we randomly select one server to use its trace to generate the workload burst, considering the burst may be caused by a certain type of workload that is normally hosted by only a few servers. We increase the request rate from the 5th minute to the (5+L)th minute by the burst degree, where L is the burst duration, to create a bursty trace. We normalize the bursty trace to the peak request rate of the aggregated trace to get the *Yahoo* trace in terms of percentage. In the simulations, we test different burst degrees (from 2.6 to 3.6) and burst durations (1, 5, 10 and 15 minutes). The *Yahoo* trace in Figure 7(b) has burst degree = 3.2 and burst duration = 15 minutes.

VII. Results

In this section, we present our results from both the simulation and the prototype hardware testbed. The computing performance of each sprinting strategy is normalized to the performance without sprinting.

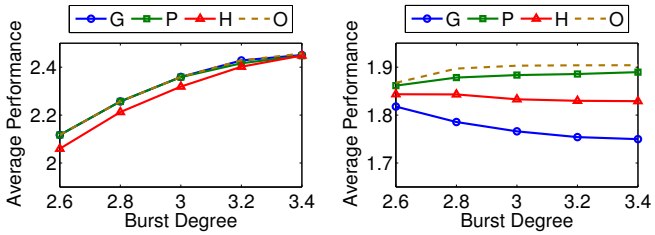
A. Comparison with Uncontrolled Sprinting

We first explain why appropriate control at the data center level (and PDU level) is necessary when conducting the chip-level sprinting. Figure 8(a) shows sprinting without DC-level control can cause the CB to trip after only 5 min 20 sec, if we simply turn on extra cores to achieve the required performance. To avoid such a disastrous consequence, we have to finish the chip-level sprinting before this moment by shutting down most cores, which results in low performance. In contrast, Data Center Sprinting allows the high performance to last for a much longer time, as shown in Figure 8(b). The additional energy is obtained by managing the ESDs and CB overload. In this case, the UPS and TES provide 54% and 13% of the additional energy on average at the PDU level and DC level, respectively. The TES energy is also utilized to enhance cooling, such that the data center is not to be overheated.

B. Comparison among Different Strategies

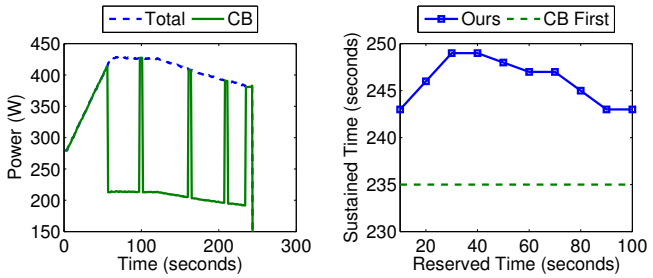
It is important to appropriately select the sprinting degree, in order to optimize the overall performance during the workload bursts. We test the four strategies that dynamically determine the upper bound of sprinting degree with the *MS* trace. We calculate the values of BDu_p (for the Prediction strategy) and SDe_p (for the Heuristic strategy) by adding an *estimation error* to the real value. For example, the Prediction strategy needs the value of BDu_p (*predicted burst duration*), and the real burst duration of the *MS* trace is 16.2 minutes (the aggregated time when the normally active cores are inadequate to handle all the workloads). We calculate BDu_p as 16.2 minutes $\times$ (1 + *estimation error*), where the estimation error can vary depending on how accurate the prediction of workload burst can be. Similarly, we get the value of SDe_p (*estimated best average sprinting degree*) for the Heuristic strategy based on the real best average sprinting degree and the estimation error. The flexibility factor K% of the Heuristic strategy is assumed to be 10% in our experiments.

The results of the four strategies are shown in Figure 9. The Prediction and Heuristic strategies are affected by the estimation error, while the Greedy and Oracle strategies need no estimation and hence are independent of the error. We can observe that when the estimation error is zero, both Prediction and Heuristic can achieve an average performance close to the oracle result. When BDu_p (the burst duration) is overestimated or SDe_p (the best average sprinting degree) is underestimated, Prediction and Heuristic are still considerably better than Greedy, because their upper bounds of sprinting degree are configured to be relatively low at the beginning. This does not degrade the power efficiency, and the upper bound can be adjusted online by their algorithms. However, if BDu_p is underestimated or SDe_p is overestimated, the upper bound of sprinting degree would be too high at the beginning, which degrades the power efficiency. Therefore, even if the upper bound is adjusted later, the overall result can be still unsatisfactory (sometimes no better than Greedy). Overall, Data Center Sprinting can improve the average performance



(a) 5-min burst duration (b) 15-min burst duration

Fig. 10. The average performance with different bursts of the *Yahoo* trace. G: Greedy; P: Prediction; H: Heuristic; O: Oracle.



(a) Reserved trip time = 10 sec (b) Compared to *CB First*

Fig. 11. (a) The power curve from the hardware testbed experiments. (b) The comparison to *CB First*, a simple combination of using the CB tolerance and the UPS energy. The reserved trip time indicates how aggressively we exploit the CB tolerance.

by a factor of 1.62 to 1.76 with the *MS* trace.

C. Impact of Burst Degree and Duration

In order to evaluate the performance of Data Center Sprinting for workload bursts with different degrees and durations, here we use the *Yahoo* trace to test the four strategies, assuming the estimation error is zero for the Prediction and Heuristic strategies.

When the burst duration is relatively short (e.g., 5 minutes, Figure 10(a)), we can observe that the Greedy strategy can achieve the same performance as the Oracle strategy. This is because the stored energy would not be exhausted within the short burst duration even if the sprinting degree is not constrained. The Prediction and Heuristic strategies can also achieve good results, while the Heuristic strategy performs a little worse because it constrains the sprinting degree by the upper bound SDe_u. SDe_u is determined based on SDe_p, which can be sometimes lower than the demanded sprinting degree, and hence SDe_u can also be unnecessarily low.

When the burst duration is relatively long (e.g., 15 minutes, Figure 10(b)) and the stored energy becomes inadequate, a higher burst degree can make the stored energy consumed in a more inefficient way if we do not constrain the sprinting degree. Therefore the performance of the Greedy strategy is significantly degraded. The Prediction and Heuristic strategies achieve better performance because of limiting the sprinting degree to extend the sprinting duration and improve the power efficiency. The Prediction strategy also performs better than the Heuristic strategy, because the former makes the sprinting duration equal to the accurately predicted burst duration, while the latter uses its estimated average sprinting degree to derive the sprinting duration, which may not match the burst duration.

Although the results in Figure 10 suggest the Prediction strategy is efficient than the Heuristic strategy for different burst degrees and durations with zero estimation error, the Heuristic strategy could be more appealing since it works better in Figure 9 when the estimation error is -100% to +60%. Overall, Data Center Sprinting can improve the average performance by a factor of 1.75 to 2.45 with the *Yahoo* trace.

D. Hardware Testbed Results

In the testbed experiments, we study how the CB and UPS contribute to the sprinting process. Our CB can sustain at most 232 W without being overloaded. When the UPS is connected with the server by the relay, it supplies about half of the power, and the CB is not overloaded even if the server power peaks (428 W, whose half is less than 232 W). By manipulating the connection between the server and the UPS, we can determine whether to overload the CB or discharge the UPS battery in each second. We do the experiments using the *Yahoo* trace to generate the server power, assuming the CPU utilization of the server is equal to the *Yahoo* trace with burst degree = 1. The upper bound of sprinting degree is determined by the Greedy strategy. Since the idle server power (273 W) is greater than the CB capacity, the testbed emulates a scenario that the sprinting process starts from the first second.

We use the parameter *reserved trip time* to indicates how aggressively we exploit the CB tolerance. For example, when the reserved trip time is 10 seconds (Figure 11(a)), we overload the CB only if the current CB tolerance can sustain the current overload for more than 10 seconds. Otherwise, we turn to the UPS to cancel the CB overload. We plot the total sustained time in Figure 11(b) to see how it changes with the increase of reserved trip time. We find that the sustained time is maximized if the CB is rarely overloaded when the server power is relatively high. For example, when the server power is more than 375 W, the CB is overloaded for 21, 8 or 35 seconds when the reserved trip time is 10, 30 or 90 seconds, respectively. As the result, the CB can sustain the sprinting process for a longer time with the 30-second reserved trip time. The reason is that the CB trip time increases much faster than the decrease of the CB overload, as shown in Figure 2. For example, when the CB overload decreases from 60% to 30% (2 times), the trip time increases from 1 minute to 4 minutes (4 times). Therefore, given the same amount of UPS energy, the server can use more energy when the CB overload is lower, and thus the sprinting duration can be longer.

In Figure 11(b), our solution is compared to a baseline solution *CB First*, which solely relies on CB overload at the beginning, and then utilizes UPS energy when the CB can no longer be overloaded. The maximum sustained time of our solution is 14 seconds longer than that of *CB First*. Without the UPS, the CB will trip in 65 seconds, which is only 26% of our sustained time. Therefore, it is beneficial to coordinate the CB overload and the usage of UPS energy, in order to extend the sprinting duration. Due to the lack of devices for experiments, we cannot control the magnitude of CB overload on our testbed, and hence cannot sufficiently

extend the sprinting duration. In a real data center, this can be realized by select a certain portion of servers to be powered by the UPS and the rest to use the power from the PDUs.

VIII. Conclusion

Computational Sprinting has been recently proposed to boost the computing performance for bursty computation demands. However, conducting sprinting in a data center faces new challenges because there are power and thermal constraints at the data center level, which are exacerbated by recently proposed power infrastructure under-provisioning and reliance on renewable energy, as well as the increasing server density. In this paper, we have proposed Data Center Sprinting, a methodology that enables a data center to temporarily boost its computing performance by turning on more cores in the era of dark silicon, in order to handle bursty workload. We demonstrate its feasibility by analyzing data center circuit breakers and energy storage devices, to realize safe sprinting without causing undesired server overheating or shutdown. We design four strategies that optimize the sprinting degree, to improve the average performance during workload bursts. We evaluate a prototype of Data Center Sprinting on a hardware testbed and in datacenter-level simulations. The experimental results show that our solution can improve the average computing performance of a data center by a factor of 1.62 to 2.45 for 5 to 30 minutes.

References

[1] [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[2] L. A. Barroso and U. Holzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool, 2009.

[3] A. Bhattacharya, A. Kansal, D. Culler, S. Sankar, and S. Govindan, "The need for speed and stability in data center power capping," in *IGCC*, 2012.

[4] S. Borkar and A. A. Chien, "The future of microprocessors," *Communications of the ACM, Vol. 54 No. 5, Pages 67-77*, 2011.

[5] J. S. Chase, D. C. Anderson, P. N. Thakar, and A. M. Vahdat, "Managing energy and server resources in hosting centers," in *SOSP*, 2001.

[6] Y. Chen, S. Jain, V. K. Adhikari, Z.-L. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via yahoo! datasets," in *Infocom*, 2011.

[7] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *ISCA*, 2011.

[8] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ISCA*, 2007.

[9] B. Forrest. [Online]. Available: http://radar.oreilly.com/2009/06/bing-and-google-agree-slow-pag.html

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

[11] D. Garday and J. Housley, "Thermal storage system provides emergency data center cooling," *Intel Information Technology on Computer Manufacturing Thermal Management*, 2007.

[12] S. Govindan, A. Sivasubramaniam, and B. Urgaonkar, "Benefits and limitations of tapping into stored energy for datacenters," in *ISCA*, 2011.

[13] S. Govindan, D. Wang, A. Sivasubramaniam, and B. Urgaonkar, "Leveraging stored energy for handling power emergencies in aggressively provisioned datacenters," in *ASPLOS*, 2012.

[14] J. Howard *et al.*, "A 48-core ia-32 message-passing processor with dvfs in 45nm cmos," in *ISSCC*, 2010.

[15] 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.

[16] M. Iyengar and R. Schmidt, "Energy consumption of information technology data centers," *Electronics Cooling*, December, 2002.

[17] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of datacenter traffic: Measurements & analysis," in *IMC*, 2009.

[18] 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.

[19] M. Lassnig, T. Fahringer, V. Garonne, A. Molfetas, and M. Branco, "Identification, modelling and prediction of non-periodic bursts in workloads," in *CCGrid*, 2010.

[20] 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.

[21] C. Li, A. Qouneh, and T. Li, "iSwitch: Coordinating and optimizing renewable energy powered server clusters," in *ISCA*, 2012.

[22] P. Lin, S. Zhang, and J. VanGilder, "Data center temperature rise during a cooling system outage," *White Paper of Schneider Electric Data Center Science Center*, 2013.

[23] 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.

[24] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," in *ISCA*, 2011.

[25] R. Miller. [Online]. Available: http://www.datacenterknowledge.com/archives/2012/04/03/google-embraces-thermal-storage-in-taiwan/

[26] ——. [Online]. Available: http://www.datacenterknowledge.com/archives/2012/05/15/google-data-center-faq/

[27] ——. [Online]. Available: http://www.datacenterknowledge.com/archives/2011/08/01/report-google-uses-about-900000-servers/

[28] J. Musilli and B. Ellison, "Facilities design for high-density data centers," *Intel White Paper*, 2012.

[29] National Fire Protection Association, 2008.

[30] S. Pelley, D. Meisner, T. Wenisch, and J. VanGilder, "Understanding and abstracting total data center power," in *WEED*, 2009.

[31] 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.

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

[33] 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.

[34] P. Ranganathan, P. Leech, D. E. Irwin, and J. S. Chase, "Ensemble-level power management for dense blade servers," in *ISCA*, 2006.

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

[36] G. Shanmuganathan, A. Gulati, A. Holler, S. Kalyanaraman, P. Padala, X. Zhu, and R. Griffith, "Towards proactive resource management in virtualized datacenters," *VMware Labs*, 2013.

[37] A. Shilov. [Online]. Available: http://www.xbitlabs.com/news/cpu/display/20050913222050.html

[38] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Transactions on Computers, 63(11): 2647-2660*, 2013.

[39] S. Tenbohlen, T. Stirl, and M. Stach, "Assessment of overload capacity of power transformers by on-line monitoring systems," in *IEEE Power Engineering Society Winter Meeting*, 2001.

[40] J. Verge. [Online]. Available: http://www.datacenterknowledge.com/archives/2013/12/03/study-cost-data-center-downtime-rising/

[41] D. Wang, C. Ren, A. Sivasubramaniam, B. Urgaonkar, and H. Fathy, "Energy storage in datacenters: What, where, and how much?" in *SIGMETRICS*, 2012.

[42] D. Wang, S. Govindan, A. Sivasubramaniam, A. Kansal, J. Liu, and B. Khessib, "Underprovisioning backup power infrastructure for datacenters," in *ASPLOS*, 2014.

[43] X. Wang, M. Chen, C. Lefurgy, and T. Keller, "SHIP: A scalable hierarchical power control architecture for large-scale data centers," in *TPDS*, 2011.

[44] 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.

[45] Y. Zhang, Y. Wang, and X. Wang, "TEStore: Exploiting thermal and energy storage to cut the electricity bill for datacenter cooling," in *CNSM*, 2012.

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