GreenSprint: Effective Computational Sprinting in Green Data Centers

Haoran Cai*, Xu Zhou†, Qiang Cao*✉, Hong Jiang§, Feng Sheng*, Xiandong Qi‡,
Jie Yao*, Changsheng Xie*, Liang Xiao*, Liang Gu†
*Wuhan National Laboratory for Optoelectronics, Key Laboratory of Information Storage System of Ministry of Education,
School of Computer Science and Technology, Huazhong University of Science and Technology
§Department of Computer Science and Engineering, University of Texas at Arlington
‡Department of Computer Science and Engineering, Hong Kong University of Science and Technology
† Sangfor Technologies Co., Ltd.
✉Corresponding Author: caoqiang@hust.edu.cn

***Abstract*—Computational Sprinting has proven to be an effective way to boost the computing performance for bursty workloads, which allows a chip to exceed its power and thermal limits temporarily by turning on all processor cores and absorbing the extra heat dissipation with certain phase-changing materials. However, extra power available for sprinting is constrained by existing power distribution infrastructures. Using batteries alone to provide the additional power to achieve performance target not only limits the effectiveness of sprinting, but also negatively impacts the lifetime of the batteries. Leveraging renewable power supply in a green data center provides an opportunity to exploit the maximal potential of Computational Sprinting. However, the intermittent nature of renewable energy makes it very challenging. In this paper, we propose GreenSprint, a renewable energy driven approach that enables a data center to boost its computing performance efficiently by conducting computational sprinting. We present four sprinting strategies to address the challenge imposed by the intermittent and time-varying nature of renewable energy supply. We build an experimental prototype to evaluate GreenSprint on a cluster of 10 servers with a simulated solar power generator. The results show that renewable energy by itself can sustain different duration lengths of sprinting when its supply is sufficient and can improve performance by up to 4.8x for representative interactive applications. We also show the effectiveness of core-count and frequency scaling in the presence of varied renewable power and limited battery energy.**

I. Introduction

Computational sprinting has been widely explored in recent years [7], [8], [30], [23], [16]. Due to thermal constraints, some of the processor cores on a chip must be powered off most of the time, a phenomenon known as dark silicon [16], [9]. Many recent studies have demonstrated that computational sprinting, in which idle cores are activated and voltage and/or frequency are increased to allow the thermal constraints to be crossed for a short period of time by absorbing the extra heat dissipation with special phase-changing materials [23], [8], can effectively and significantly speed up application performance during workload bursts. For data centers with interactive workloads (e.g., search, forum, news), while workload bursts can be less frequent, the intensity of such bursts are usually much higher under a variety of circumstances [30], such as breaking news, online shopping big sales (e.g., the Black Friday after Thanksgiving), etc. As illustrated by Figure 1, the diurnal workload pattern (dotted line) from a study of a Google data center [13] consists of several load spikes during the whole day with varying burst intensities and durations. There exists a great opportunity for computational sprinting to guarantee the quality of service in these cases.

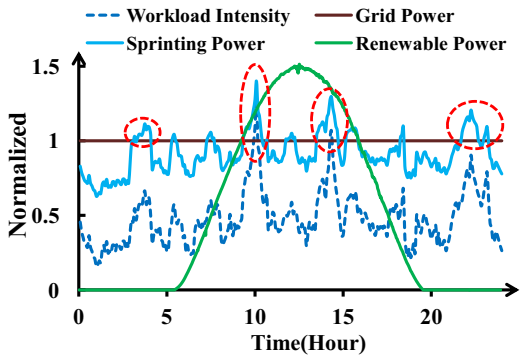


Fig. 1: Workload pattern for a Google data center [13] and scaled power demand of sprinting normalized to grid power.

However, many prior works demonstrated that today's data centers are already approaching the peak capacity of their power infrastructures [10], [15] which is similar to the thermal constraint at the chip level. The extra bursty power demand required by computational sprinting at the datacenter level can lead to serious power emergencies [30] as indicated in Figure 1 by the red ovals when the demand exceeds the grid power capacity. Rejecting service requests due to power capacity cap may cause data centers to lose revenue and customers in the long term. Existing solutions to deal with bursty power demand mainly focus on battery-backed power system [25], [12] or one combined with overloading circuit breaker [30], [23]. However, using battery alone can be energy inefficient and harms the lifetime of batteries due to the frequent charging/discharging activities [18]. An emerging solution to the power emergency problem is to leverage green energy sources to supplement grid power capacity. In such green data centers, power-constrained grid can be used as backup for the renewable power supply or vice versa. Also,

1530-2075/18/$31.00 ©2018 IEEE
DOI 10.1109/IPDPS.2018.00078

renewable power solution can tackle the environmental challenges brought by power consumption and carbon emissions. Compared with traditional data centers without green power supply, this solution avoids expensive capital expenditure and time-consuming construction cycle (ranging in the hundreds of millions USD and decades of years) of upgrading grid power infrastructures [26], [31], [27]. In a typical data center, the capital cost spent on provisioning the grid utility infrastructure is between $10-$25 for each watt [31]. Therefore, the renewable power solution brings us an opportunity to deal with burst power demand in a cost-efficient way.

Thus, in this paper, we first ask and then try to answer the following questions, *can we leverage green power supply to support computational sprinting in green data centers, and if so, how?* Although renewable energy is an attractive solution, it is also well known for its intermittent and variable nature which is also demonstrated in Figure 1 for a typical solar power supply. Thus, directly conducting sprinting using renewable energy in a green data center, without appropriate control, can have negative impacts, for example, reducing the lifetime of batteries, degrading the performance of services leading to SLA violations, and even causing power failures in the data center. Hence, we split the solution into three cases: (1) When the green power supply complementing the grid power can fully satisfy the bursty power requirement, we operate the sprinting with only green power and then charge the surplus green power into energy storage devices. (2) When the green power supply is insufficient for the power demand, the batteries, strategically charged either by renewable source or grid source during non-sprinting periods, discharge to make up for the power shortage. (3) When the green power is not available, certain power management knobs (e.g., server-level low power state) can be adopted to manage sprinting power to match the current power provision, potentially resulting in a performance degradation for some applications. Obviously, applying computational sprinting by leveraging green power to provision power bursts in power-constrained data centers can help significantly save the capital expense while improving application performance.

Based on the analysis above, we propose GreenSprint, a green data center based approach that exploits renewable energy to effectively and efficiently conduct computational sprinting. To the best of our knowledge, while many prior works have focused on supporting computational sprinting at the chip level or at the data center level by utilizing battery supply only, the issues of computational sprinting in the presence of green energy at the data center level and its cost-benefit trade-offs have not been well addressed in the literature. In exploring the design space of employing renewable energy for computational sprinting, this paper makes the following contributions: (1) We propose *GreenSprint*, a renewable energy driven approach that enables data center level sprinting by turning on more cores and boosting their voltage and frequency in the era of dark silicon, in order to handle occasional workload bursts. (2) We present four strategies to determine the core count and the frequency level for sprinting based on the intermittent and time-varying renewable power supply. Especially, we propose a *Hybrid* strategy that combines reinforcement learning to determine the optimal server setting, targeting at the power provision safety and the quality of service. (3) We develop an experimental prototype consisting of 10 servers, a simulated solar power generator, and a server-level battery provision to evaluate our approach. Using representative data center workloads, the evaluation shows that our solution can improve the average computing performance by up to 4.8x for SPECjbb, 4.1x for Web-Search, and 4.7x for Memcached with renewable power supply. (4) Based on the evaluation results, we present specific analysis on the interplay between renewable power, battery energy, sprinting duration, workload characteristics. We draw several insightful observations to guide computational sprinting in green data centers.

II. Power Infrastructure in Green Data centers

Considering the fact only part of the cores in the multicore servers in typical data centers are active due to the dark silicon phenomenon, it offers the potential to apply the computational sprinting technique to boost performance of applications with bursty workloads, particularly interactive workloads, by turning on additional cores and increasing their voltage and frequency. This is possible, however, only if thermal constraints at both the server/chip level and data center level can be temporarily stretched, i.e., with the necessary heat-absorbing materials and cooling equipped, and the power supply infrastructure is able to meet the bursty power demand of sprinting whenever the demand arises. Since our focus in this paper is on leveraging renewable energy in data centers to meet the bursty power demand of computational sprinting, in this section we will introduce the power infrastructure for computational sprinting in a green data center and make appropriate assumptions about the thermal constraints.

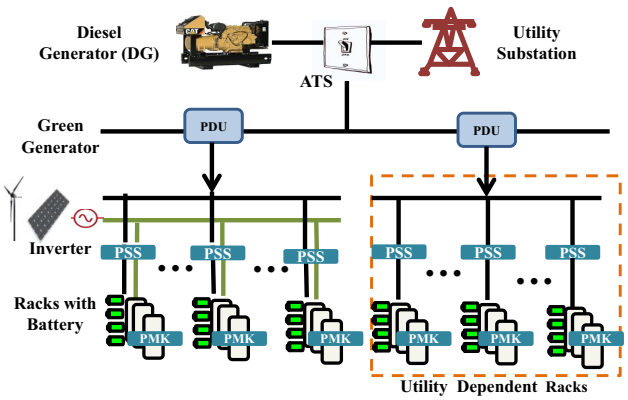


Fig. 2: The Power Infrastructure of GreenSprint

Overview: Figure 2 depicts an architectural overview of an on-site green data center power hierarchy for computational sprinting, similar to prior works [32], [10]. To achieve the sprinting goal, we directly connect the on-site renewable power supplies such as photovoltaic (PV) and wind to the power

distribution unit (PDU) level to provide a dual-power supply of the grid and renewable power rather than integrating the renewable energy into the utility power. This can help decrease the impacts of voltage transients, frequency distortions and harmonics. Compared with the centralized power integration, our distributed integration prevents PDUs becoming a power delivery bottleneck. The existing uniform centralized power provision mechanism forces all servers to obtain the same or similar renewable power capacity, significantly limiting the sprinting power supply. To this end, provisioning renewable energy on the PDU level allows us to apply computational sprinting in a data center on a per-rack basis. When we conduct sprinting, some servers will be powered only by the renewable energy with a separate green power bus while others will depend on the grid power. This can help to greatly relieve the burden on the circuit breakers (CB) and other constrained power infrastructure.

Renewable power: The greatest challenge facing powering sprinting with renewable energy is its time-varying, intermittent power. Therefore, we employ a power-source selector (PSS) to adaptively switch among different power sources (i.e., green, battery and grid power). PSS performs switch tuning based on the discrepancy between the workload power demand and the green power supply. PSS is also configured to charge the battery when there is an excess of green power, and discharge it when green power is insufficient or unavailable. PSS can identify the switching parameters for the inverter and charge controllers of batteries to allow for a full control of every power source. Programmable power electronics circuitry can be used to implement PSS. However, since our experimental setup does not provide this functionality, our evaluations account for such control capability in the form of managing the sprinting decisions. As shown in Figure 2, a server-level power management knob (PMK) receives the execution output from the PSS to control the power demand on a per-server basis. When renewable power and battery are not sufficient, PMK decreases the sprinting intensity to keep servers within the power budget by considering applications' diverse characteristics.

Battery: Energy storage devices must be deployed to support sprinting continuously when the renewable energy is insufficient. Prior works proposed to rely on uninterruptible power supply (UPS) devices when the utility power source suddenly fails [23], [30]. Since we connect the green power to the PDU level, we also leverage the distributed battery architecture shown in Figure 2, which is widely employed by IT companies such as Google [1] (server-level battery) and Facebook [3] (rack-level) to smooth the supply of the renewable power. The former design achieves energy efficiency by bringing the AC distribution (green and grid) even closer to the IT load, before it is converted (we adopt this design in our solution). The distributed design can provide great scalability and avoid AC-DC-AC double conversion.

However, battery-based sprinting can have significant adversary effects on the battery lifetime because batteries can wear out under irregular charging and discharging regimes [11]. There are two main factors affecting the battery failure rate. First, discharging current, which has strong relationship with the sprinting intensity, indicates the capacity performance of battery. For example, while the rated capacity is 24Ah at a 20-hour discharging rate, the capacity drops to only 12Ah at a 12-min discharging rate. Second, high depth of discharging (DoD) can degrade the battery lifetime. For the purpose of prolonging the battery lifetime, we cannot exhaust the energy of a battery. In our work, we model a server-level 12V value-regulated lead-acid battery (VRLA), similar to that used by Li et al. [11]. Batteries are characterized by their supply time as approximated by Peukert's Law (Peukert's exponent is 1.15 for LA battery [27]), which shows the time taken to drain a certain capacity for different power demands. We also assume DoD=40% in our setup, which translates to a lifetime of 1300 recharge cycles [27].

Thermal concerns at the chip level: Another challenge is cooling at the server level. A chip multiprocessor's sprinting level depends on its thermal package and heat sink. In other words, the thermal constraint at the chip level directly determines the maximum duration of sprinting. Sprinting activities produce more heat into the ambient environment than in normal server mode. Therefore, server system needs to effectively remove the extra heat and prevents over-heating on chip-level. Fortunately, prior work [21] found an effective way to shape the thermal load of a data center. In that work, phase changing materials (PCM) is used to temporarily store the heat generated by servers and other equipment during peak loads, and release the heat when there is excess cooling capacity during non-sprinting periods. It shows that PCM can delay the onset of thermal limits by hours. In our work, we assume servers are equipped with such thermal package for sprinting and the server can sustain different demands of heat dissipation resulting from computational sprinting.

III. Design of GreenSprint

In this section, we present GreenSprint, a renewable energy driven framework that enables data center sprinting by managing power sources and scheduling workloads. The GreenSprint framework is designed to assist the power source selector (PSS) and power management knobs (PMK) in their decision-making process. We present the architecture of GreenSprint in Figure 3. The main components are *Monitor*, *Predictor*, *PSS*, and *PMK*. The *Monitor* collects the performance of workload (e.g., latency and throughput) and the power used (e.g., battery energy, renewable power, and server power). The *Predictor* predicts the workload intensity and the renewable energy production. The *PSS* takes these predictions and the available power to choose appropriate power sources. The *PMK* uses the profiling records and the power supply to manage sprinting activities for busty workloads. In the following, we emphasize on the details of the *PSS* and *PMK*.

A. Power Source Selector

The sprinting power provision can come from renewable power or battery, depending on the decision made by a power

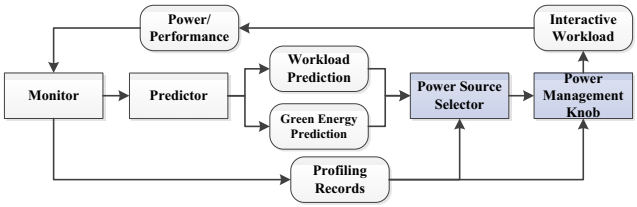


Fig. 3: The GreenSprint Architecture

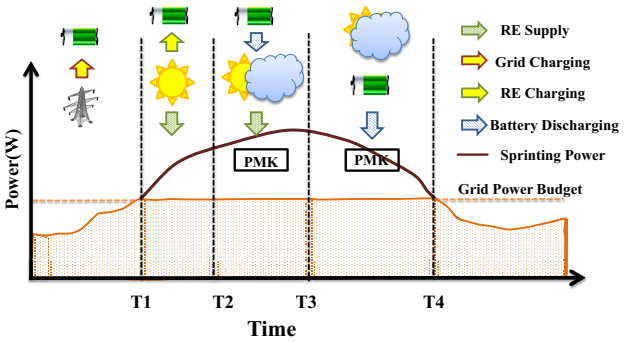


Fig. 4: An Illustration of Power Source Selector under Different Power Supply/Demand Scenarios

source selector (PSS). Figure 4 illustrates how the green power and battery energy are provided at the rack level. In each case, the duration of a power burst is divided into a series of discrete scheduling epochs ($T1$, $T2$, ..., Tn) which can be classified into the following three possible cases:

Case 1: Renewable power, (*RESupp*), is abundant and can be independently used for sprinting (from T1 to T2). The excess power beyond the sprinting needs can be used to charge the battery. In this period, power supply depends on renewable power. Sprinting starts from additional cores being activated and ends when the workload requests are finished or batteries join back in power supply.

Case 2: Renewable power is insufficient. Due to the time-varying, intermittent nature (e.g., weather condition, time of sprinting, etc.), the green power supply temporarily needs the supplement from other power sources, such as batteries. To this end, we employ battery power (*BattSupp*) to supplement the green power to sustain the sprinting (from T2 to T3) immediately. To make this work, power management knobs (PMK) must work cooperatively with PSS. This case ends when green power supply becomes unavailable.

Case 3: The battery independently sustains the sprinting when the renewable power is unavailable (from T3 to T4). If the workload burst can be completed in this period, then we charge the battery with grid power in anticipation of future sprints. The worst case happens when there is no sufficient battery energy left, then overloading circuit breaker (CB) for the grid power may be the last resort to maintaining sprinting. Recharging is activated when battery depth of discharge reaches the set goal (40% DoD). To prevent tripping the CBs with too much power overload, we limit the total power of all the downstream branches under an upper bound. Again, due to limitations on batteries, PMK and PSS should cooperate with each other in case of a power emergency.

To help the PSS determine which case the power system should step in, GreenSprint makes a judgement on the relationship between *RESupp*, *BattSupp* and the power demand *Load-Power* in each scheduling epoch. In this design, we calculate *BattSupp* based on previous discharging activities. Specifically, the energy usage of battery during each epoch is recorded by a controller node in a cluster. The available capacity is derived from the maximum capacity and the used capacity. To properly capture *Peukert's* effect during discharging, we recalculate the remaining discharging time after each scheduling epoch.

For *RESupp*, we present a renewable energy prediction model for the *Predictor* with lower complexity and short time horizons. The prediction is based on the past power production records. In particular, we continuously calculate an exponentially weighted moving average (EWMA) of the past average power production in Equation 1:

$$RESupp(t) = \alpha * RESupp(t-1) + (1-\alpha) * Obs(t) \quad (1)$$

where *Obs(t)* is the observed power production and *RESupp(t)* is the predicted power supply for the next epoch (e.g. of 5 minutes) in the current epoch *t*. α reflects a tradeoff between stability and responsiveness and it ranges from 0 to 1. When α varies, we find α=0.3 to be the most consistent, which weights the model more heavily towards current observed data. Note that most solar prediction algorithms are accurate when weather conditions are stable.

B. Power Management Knobs

The power management knob (PMK) is introduced to manage power demand. When a workload burst occurs and there is an abundant renewable power supply, the workload requests can be quickly completed by sprinting without triggering the grid or battery power. Further, the surplus renewable energy is used to charge the battery. When the power source can no longer sustain the power demand, we finish sprinting by deactivating the additional active cores and setting the frequency to the lowest level, *Normal* mode. The case where the power supply is insufficient becomes more complicated. PMK should determine an appropriate sprinting intensity for better energy efficiency. In what follows, we present four different power management strategies for computational sprinting in a green data center, namely, *Greedy*, *Parallel*, *Pacing* and *Hybrid*.

The most straightforward solution is to simply activate all cores and set the highest frequency to temporarily accommodate workload bursts. We call this the *Greedy* strategy because it needs aggressive power supply. This strategy does not assume any prediction of the future green energy production. It greedily tries to run with the maximal sprinting intensity, seeking maximum improvement on performance for each application. When the power supply is sufficient or the workload intensity is moderate, *Greedy* can strictly ensure the quality of service (QoS) for latency-critical applications, further reduce the response time of each request. For example,

Greedy can achieve an average 270ms latency for SPECjbb at 70% burst load intensity, while a best-efficiency policy (lower sprinting intensity and higher response time) can only provide 466ms latency with a 500ms latency constraint. Obviously, most service providers are willing to deploy *Greedy* that improves the user experience by maximal sprinting, which may bring additional revenue. However, when the power supply is insufficient, or the workload bursts are intensive, the service may be unavailable due the high power consumption of *Greedy*. When the batteries kick in, considering the burst duration, lower sprinting intensity may be a better choice because of lower power consumption, which leads to longer discharging time. For comparison, we develop the following three strategies.

During each sprinting interval, each server $j = 1, ..., n$ in a rack c can operate in a particular sprinting intensity $S_j \in S$. S is a two-dimensional set consisting of frequency level and core count. It is ordered from S_0, which is the *Normal* mode (e.g., 6 cores with 1.2 GHz in our testbed), to S_r, which is the maximum sprinting mode (e.g., 12 cores with 2.0 GHz). We also denote the intensity of a workload during this sprinting interval as $L_j \in L$, which can be any of the w levels, $L_1, ..., L_w$, between the minimum and maximum intensity levels for a given application. The power demand for each sprinting epoch t not only depends on the workload intensity level $L_{j,t}$ being served on server j during epoch t but also on the server configuration S_j. We measure and collect the power demand (denoted as $LoadPower_j(L_{j,t}, S_{j,t})$) of an individual workload for each server settings S_j and workload intensity levels L_j with *a priori* knowledge using an exhaustive method on real servers.

Another two strategies we proposed are called *Parallel* and *Pacing*. *Parallel* scales only the core count while *Pacing* scales only the frequency levels at each time. For *Parallel* and *Pacing*, we intend to explore the impact of two power scaling techniques on the performance with the renewable power supply, which has not been well discussed in prior works. We use the EWMA prediction again to predict the workload intensity $L_{pre,t}$. Then the potential maximal power demand can be denoted as $LoadPower_j(L_{pre,t}, S_{r,t})$. Then there exists a power mismatch between power supply and demand, denoted as $M_t = \sum_{j=1}^{n} LoadPower_j(L_{pre,t}, S_{r,t})$-$PowerSupp_t$, where $PowerSupp_t$ is the sum of $RESupp(t)$ and $BattSupp(t)$ for the whole rack. The power reduction $P_{M,t} = \sum_{j=1}^{n}(LoadPower_j(L_{pre,t}, S_{r,t}) - LoadPower_j(L_{pre,t}, S_{j,t}))$ offered by scaling core count or frequency level for each epoch t. Therefore, we handle the power mismatch M_t by carefully managing power reduction $P_{M,t}$. Thus, we arrive at the following equation:

$$\forall t \in T : RESupp(t) + PBattSupp(t) + P_{M,t} = \sum_{j=1}^{n} LoadPower_j(L_{pre,t}, S_{r,t}) \quad (2)$$

To this end, an optimal setting S_j is achievable to maximize the overall performance $Perf_{j,t}$ in each epoch t for servers in rack c. We denote the optimization target as:

$$max \sum^{t \in T} \sum^{j \in c} Perf_{j,t}(L_{j,t}, S_{j,t}) \quad (3)$$

Parallel and *Pacing* solve the problem under constraints on service quality of service (QoS), renewable power production, and battery power supply (e.g., DoD, capacity), which is similar to some previous studies [25], [32], [20].

Algorithm 1 Reward mechanism

```
// Calculate reward r_t based on epoch t and t+1
QoS_target and QoS_current represent the target QoS of the workload and the current latency result. PowerSupp and PowerCurr are the power supply and the current power demand at time t.
R_power = PowerSupp / PowerCurr      //Power reward
R_qos = QoS_target / QoS_current     //QoS reward
If R_power > 1 Then
    If R_qos > 1 Then
        r_t = R_power+R_qos+1
    Else
        r_t = R_power-R_qos+1
    EndIf
Else
    r_t = -R_power-1
EndIf
// Update the value of R(c_t, a_t) in lookup table
R(c_t, a_t)=R(c_t, a_t) + α[r_t + γ max_{a_i ∈ S} R(c_{t+1}, a_i) - R(c_t, a_t)]
```

Finally, we present a *Hybrid* strategy, which combines both frequency and core count scaling in *Parallel* and *Pacing* with *reinforcement learning*. *Hybrid* tries to learn the optimal settings to achieve higher energy efficiency and strict QoS guarantee. In our work, we first fomulate this problem as a Markov Decision Process (MDP). In an MDP, a decision-making process must learn the best course of action to maximize its total reward over time. At each discrete epoch, the system can observe its current *state*, c_t, and it must choose an *action*, a_t from a finite set of alternatives. Depending on the chosen action and current state, there is an unknown probability distribution controlling which state c_{t+1} it enters next and the reward r_t that it receives. The problem is to maximize the total discounted reward, $\sum_{t=0}^{T} \gamma^t r_t$, where γ is the discounting factor. γ should be positive and less than one, in order to reflect a preference for rewards in the near future.

In our power management problem, the *state* c_t indicates the current power supply $PowerSupp$ and workload intensity, measured during epoch t-1. Specifically, we quantize the power supply for each server, from the point of idle server power to the point of maximum sprinting power, into discrete sets by static *step* like the workload intensity level L. A small step improves the energy savings, but it tends to cause frequent changes in configuration for small changes in workload intensity and power supply. In our design, we empirically determine the step as 5% to improve energy efficiency. The *action* a_t, which is chosen depending on the state, is the combinations of core count and frequency levels, i.e. $a_t \in S$. The reward r_t is determined by the level of QoS relative to the target and the power consumption relative to the power demand.

Reinforcement learning is a type of unsupervised machine learning with a focus on online learning [28]. It solves an MDP

Configurations	RE	Batt. (Server level)
RE-Batt	30% servers	10Ah
REOnly	30% servers	0
RE-SBatt	30% servers	3.2Ah
SRE-SBatt	20% servers	3.2Ah

TABLE I: Options for green provision

by maintaining a *lookup table R(c,a)*, which is similar to another work [22]. The entry estimates the total discounted reward that will be received if the action a has been chosen based on the current state c. In our work, to reduce the complexity of the problem, we learn the initial values of lookup table from the profiling data collected by *Parallel* and *Pacing* using Algorithm 1. The power reward and QoS reward in reward mechanism are defined as R_{power} and R_{qos} respectively. If R_{power} is greater than one, then the power demand has been satisfied by the power supply, it demonstrates that the server can be powered normally and the power demand has been well managed. In this case, if the QoS has been ensured, i.e. R_{qos} is greater than one, then we give a positive reward. A larger reward means sprinting can provide lower response time for each request. If the QoS can not been ensured, we add a negative reward. Finally, if R_{power} is less than one, then the power supply can not meet the demand due to sprinting, therefore the total reward is negative. Once the reward r_t has been calculated, line 15 updates the value of $R(c_t, a_t)$ in the lookup table. We empirically set the discounting factor γ as 0.9 to allow a balance between short-term and future rewards. The learning rate α, we used α=0.7 in our experiments, controls the rate at which the values of $R(c_t, a_t)$ are updated. A large value of α means that the algorithm learns quickly. *Hybrid* uses the lookup table to select the best action a_t, which is the one that gives the largest total reward; i.e. a_t = arg max$_{a \in S}$*R(c$_t$,a)*. In order to improve the decisions, we also continue to update the values in the lookup table. *Hybrid* has a simple algorithm implemented using *Python*, so its runtime overhead is negligible (<2ms).

IV. Prototype Evaluation

Our scale-down experimental prototype of GreenSprint uses a cluster of $N = 10$ servers each with two 6-core 2.0GHz Intel Xeon E5-2620 processors (i.e., 12 cores per server), 48GB RAM and 1Gbps Ethernet interface and run our applications hosted on the Ubuntu Linux OS. The cluster has an NFS storage volume shared by all the servers. The power consumption of each server is monitored by an external power meter [2]. Their idle power is around 76W. The dynamic power consumption can be modulated with 9 frequency states and sprinting scales the core count from 6 to 12.

To simulate a data center with renewable energy provision, we randomly choose one of the renewable power production traces with one-week duration from NREL [6], including irradiation every minute, and replay the chosen trace on our prototype. We scale the solar power production to correspond the power source configuration (Table I) to simulate the available renewable power output. In this table, for example, 'RE' represents renewable energy provision.

Workloads	Memory Usage	Performance Metric
SPECjbb	10GB	jops (99%-ile 500ms constrained)
Web-search	20GB	ops (90%-ile 500ms constrained)
Memcached	20GB	rps (95%-ile 10ms constrained)

TABLE II: Workload Description

'S' represents *small* renewable energy and battery energy capacity provisions. In our setup, we consider a solar panel provisioned for a server j with 275W DC output (theoretical peak power), which is in line with the existing capacities in Grapesolar [4]. Hence, we can obtain the peak renewable power AC supply for a single solar panel that generates $Peak_{RE} * \alpha = 275 * 0.77 = 211.75W$. As shown in Figure 5, for the *RE-Batt* configuration, we assume that 3 servers in our prototype are provided with a renewable energy system that is capable of supplying the maximum green power of 635.25 W. For the configuration with *SRE* that provides 3 servers with smaller renewable power supply, the maximum green power obtainable is 423.5W. We assume that each server in the cluster is equipped with a battery unit and the battery energy capacity is shown in Table I. We use *cpufreq* to scale frequency and *taskset* to redirect workload threads to right cores.

Workloads and Strategies: We consider the following representative data center workloads (Table II) that exhibit different performance characteristics with different peak power demands on renewable energy and batteries. These interactive applications are SPECjbb [5], an in-memory key-value store Memcached benchmark, and Web-search from Cloudsuite [19]. We measure the maximal sprinting power demand of each application, yielding 155W for SPECjbb, 156W for Web-Search and 146W for Memcached. We also evaluate five strategies for comparison. They are *Normal*, *Greedy*, *Parallel*, *Pacing*, and *Hybrid*.

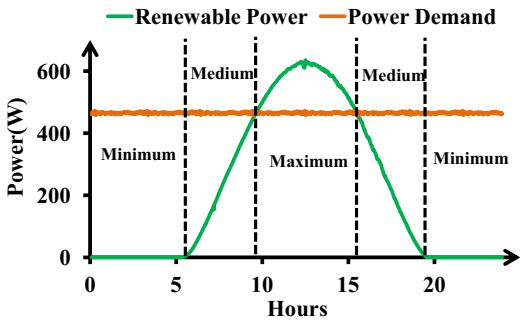
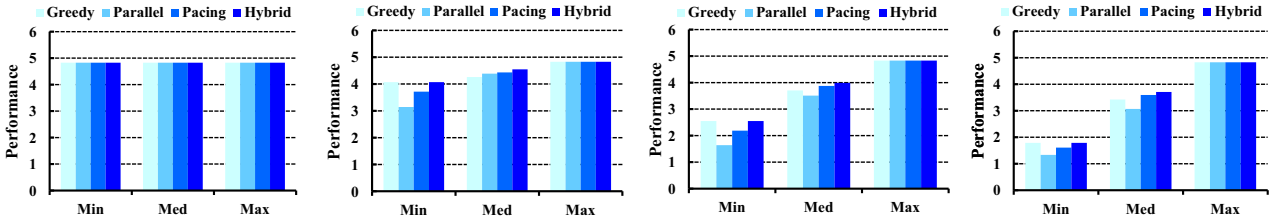


Fig. 5: The SPECjbb power profile as a function of the renewable energy availability over time.

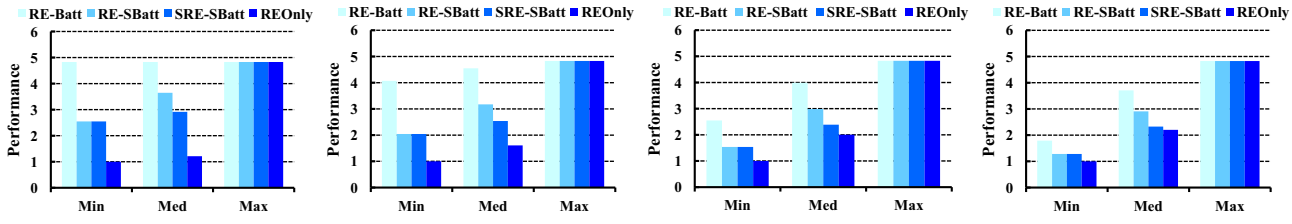
A. Performance of GreenSprint

We now compare the performance and power impact of GreenSprint against the baseline (i.e., without renewable energy). We first show representative results for SPECjbb. We generate the workload in the cluster until all 10 servers are fully utilized to produce a workload burst. As a result, the aggregate power draw of these servers can exceed the grid power budget. We statically set the sprinting to the highest



(a) 10 Mins (b) 15 Mins (c) 30 Mins (d) 60 Mins

Fig. 6: Performance of GreenSprint with varying renewable energy availability and burst durations for SPECjbb using RE-Batt, normalized to *Normal*.



(a) 10 Mins (b) 15 Mins (c) 30 Mins (d) 60 Mins

Fig. 7: Performance of GreenSprint for SPECjbb using different power configurations, normalized to *Normal*.

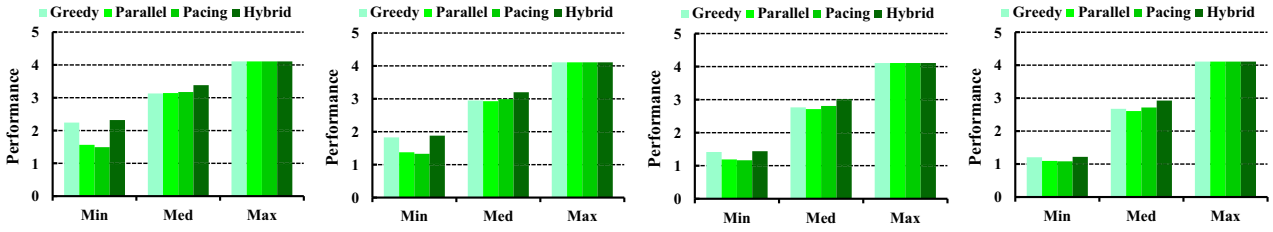
intensity. For instance, when the workload saturates all 10 servers with 12 active cores, the aggregate power consumption hits 1550W. If the grid power infrastructure can support 10 servers to operate at *Normal* mode, then the power budget of the grid can be 1000W. From the renewable energy side, if renewable energy can supply 3 servers in the cluster (i.e., RE-Batt configuration), then the grid can conservatively support the other 7 servers sprinting at sub-optimal performance (e.g., 12 core-sprinting with 1.5GHz or 7 core-sprinting with 2GHz). As specified above, we inject the workloads to deliberately induce power burst durations of 10, 15, 30 and 60 minutes. We use the average throughput (jops) of whole cluster as our performance metric for SPECjbb. To find out the impact of renewable power supply, we mainly focus on the analysis of green-provisioned servers.

Figure 5 shows the evolution of the aggregated peak power of the 3 green-provisioned servers running SPECjbb at given different levels of renewable energy *availability*. We see high variation of the renewable power production over time. We have evaluated such performance consequences for all the cases of *medium* availability over different power shortage durations. Moreover, we consider the *minimum* availability case for comparison where the sprinting goal can only be achieved by the batteries.

Impact of renewable energy availability and burst duration: Figure 6 presents the average performance of SPECjbb under different renewable energy availability and power burst durations using the RE-Batt configuration. As shown in this figure, for the *maximum* availability of renewable energy, three servers in the cluster can be directly powered by renewable energy with full-sprinting and the performance is always the best with 4.8x gains over *Normal*. Further, the surplus green power can be used to charge the battery for later use.

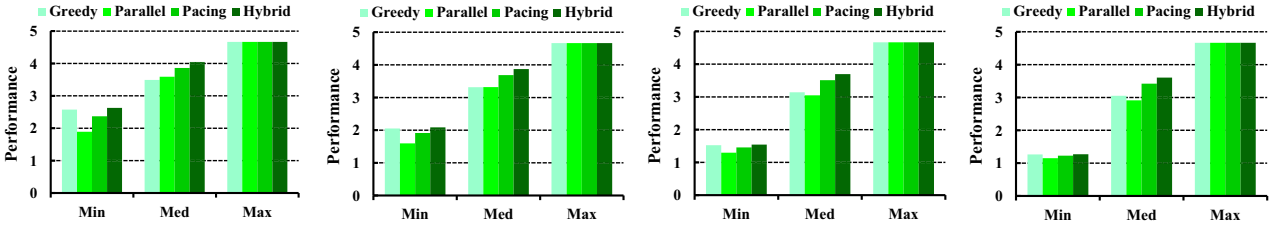
In the case of minimum and medium availability levels of renewable energy, the performance varies with different lengths of burst duration. For short bursts (10-minute duration), even when the renewable energy is unavailable, battery alone is able to completely handle the sprinting operation with maximal performance. For the durations of 15, 30, and 60 minutes, the performance varies significantly for different strategies. The performance improvement decreases relatively for longer burst durations, especially for the *minimum* availability (60-minute), in which the performance improvement drops to 1.8x for *Parallel*. Comparing with the 4.8x improvement with sufficient renewable power supply, battery-based sprinting is unsatisfactory. However, for *medium* availability, battery can supplement the green power to sustain the sprinting performance. For 60-minute durations, Sprinting can still provide up to 3.4x performance gains over *Normal*.

Impact of power management knobs: We adopt two techniques in power management knobs, scaling core counts (*Parallel*) and scaling frequency (*Pacing*). In these figures, *Pacing* slight outperforms *Parallel* in all cases. We attribute these results to higher energy efficiency of frequency scaling. Also, it demonstrates that decreasing the number of cores can still influence the performance after mitigating the oversubscription on cores. For the *Greedy* strategy with battery-based power supply, the system achieve the same results as *Hybrid* that always performs the best. This indicates that sprinting as much as possible for SPECjbb using battery receives much better energy efficiency. However, due to the higher start point of power needed to wake up servers, *Greedy* underperforms *Pacing* because it loses the opportunity to utilize the lower green power supply periods. *Hybrid* always performs the best because it always learns the optimal combinations of scaling core count and frequency for better performance.



(a) 10 Mins (b) 15 Mins (c) 30 Mins (d) 60 Mins

Fig. 8: Performance of GreenSprint for Web-Search with RE-SBatt, normalized to *Normal*.



(a) 10 Mins (b) 15 Mins (c) 30 Mins (d) 60 Mins

Fig. 9: Performance of GreenSprint for Memcached with RE-SBatt, normalized to *Normal*.

B. Impact of green configurations

We also evaluate the other three green configurations (RE-SBatt, SRE-SBatt, and REOnly) using SPECjbb. We only show the result of the *Hybrid* strategy to examine the differences among these configurations in Figure 7.

Impact of renewable energy: Configurations with RE-SBatt and SRE-SBatt show the difference of renewable power supply. When we use smaller green power, the performance degrades accordingly. However, powering two or three servers with green energy has a great effect on the cap-ex cost. Maximal performance can always be achieved during maximum green power supply. In the REOnly configuration, the performance results with minimum renewable energy availability are the same as the *Normal* mode because there is no power supply for sprinting, when all servers return to the *Normal* mode powered by the grid utility. With only renewable energy supply, GreenSprint significantly improves performance, from 2.2x (medium availability) to 4.8x (maximum availability) for the 60-minute long power burst.

Impact of battery: Given the minimum renewable energy availability, in configurations with the battery (RE-SBatt and RE-Batt) and with the minimum renewable energy availability, performance impact for the REOnly configuration can be reduced since the battery can supply additional power. Therefore, battery is preferred as a complement to deal with bursty power demand when there is no green power supply. We notice that configurations with small battery capacity (RE-SBatt and SRE-SBatt) show less improvement than the REOnly configuration when sprinting lasts for 60 minutes or longer, because the battery is not able to sustain such long operations for the entire sprinting duration. For different capacity of battery, RE-Batt (10Ah) performs better than RE-SBatt (3.2Ah) for the minimum and medium green power availability and can sustain more than 10 minutes at the maximal power burst. This implies that we can significantly improve the performance, irrespective of renewable energy availability, by purchasing a larger capacity of battery.

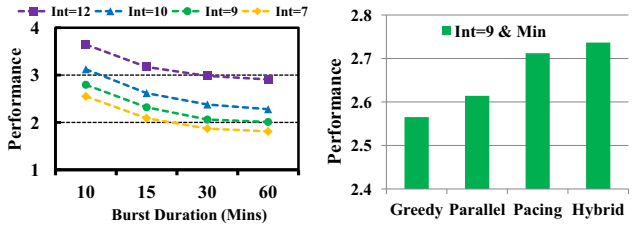
C. Impact of application characteristics

We also evaluate another two applications, Web-Search and Memcached under the RE-SBatt configuration.

Web-Search: Web-Search is the query serving portion of a production web search service and has high memory footprint. It is fairly compute intensive due to scoring and sorting search hits. In Figure 8, *Pacing* shows no more benefits than *Parallel*, relative to SPECjbb, and similar performance under varied conditions. Especially, when the green energy availability is *minimum*, lowering core count from 12 to 6 is slightly better in performance than decreasing frequency. Therefore, scaling core count can be a better choice in a battery-based power system for Web-Search because lowering chip frequency has a great impact on throughput. For longer durations, battery-based sprinting can barely achieve performance improvement over the *Normal* mode.

When the renewable energy supply is sufficient, GreenSprint can achieve 4.1x performance gain over the baseline. In face of reduction in green power, batteries can help sustain sprinting. In the *medium* case, although scaling frequency can significantly affect the performance of Web-Search, it achieves higher energy efficiency than scaling core counts, resulting in better performance. Different from constrained energy capacity of a battery system, the green power is time-varying, resulting in dynamically adjusted server power demand.

Memcached: Memcached is used as a caching service in the back-ends and loads the memory with the necessary data from disk before handling client requests. In Figure 9, the maximal performance improvement for Memcached is 4.7x, similar to Web-Search. For the *medium* and *maximum* green supply, the results show a similar trend to SPECjbb. *Pacing* performs better under different cases because of the



(a) Impact on burst durations (b) Impact on strategies

Fig. 10: Performance impact of workload burst intensity, normalized to *Normal*.

characteristics of Memcached, i.e., less computation intensive and need more on parallelism. *Greedy* is no more beneficial for both Web-Search and Memcached under battery-based supply because the optimal energy efficiency points are not always achieved with the maximal sprinting intensity.

D. Impact of workload burst intensity

To evaluate the performance of GreenSprint for different burst intensities, we generate the workload of SPECjbb by several other burst patterns to draw power bursts. In Figure 10, for example, 'Int=9' indicates the case that the peak burst load is the maximal processing capability of running workloads on 9 cores at 2.0 GHz. Figure 10(a) shows the case of *RE-SBatt* configuration and *medium* availability with *Hybrid*. According to the results, the performance is much lower (from 3.6x to 2.6x) when the burst intensity decreases (from Int=12 to Int=7) for different burst durations. Obviously, sprinting may lose the advantage on performance when burst intensity is low. In the case 'Int=7', GreenSprint can only provide 2.6x-1.7x improvement with the duration from 10 minutes to 60 minutes compared with *Normal*. Figure 10(b) presents the performance of four strategies with 'Int=9' and *minimum* availability. The duration is 10 minutes. In this case, *Greedy* performs the worst because, when the burst intensity becomes lower, maximal sprinting on 12 cores is less efficient than other strategies. Lower sprinting intensity, though higher response time under QoS constraint, can extend the discharging time of battery to achieve better overall throughput.

E. Summary of Observations

In summary, we find that: (1) Sprinting can significantly improve performance by activating more cores. (2) Despite of the intermittent nature, renewable energy can effectively support computational sprinting in power constrained data centers. (3) Batteries alone can achieve performance improvement for short sprinting durations. However, batteries are not appropriate for longer durations. (4) Renewable energy can supplement battery to reduce performance impact. (5) Compared with scaling core count for interactive applications, scaling frequency is more energy efficient in utilizing the battery energy. In other words, those interactive applications have high expectations of parallelism, so scaling core count is not the best choice to get better energy efficiency. (6) Sprinting in turn can increase the renewable power utilization due to higher power demand and energy efficiency.

F. TCO Consideration

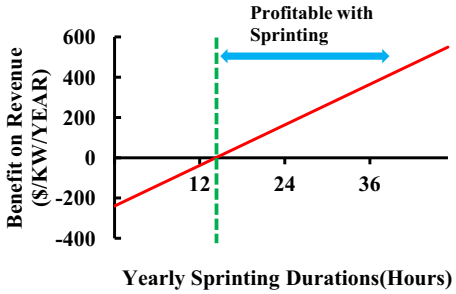


Fig. 11: POI with additional renewable energy, battery and cooling investment

Although we have considered sprinting activities with the support of renewable energy and battery that already exist in green data centers, we will illustrate whether the cost of additional green provision can be justified. Thus, we consider the revenue increase due to sprinting for two reasons. First, sprinting raises the power demand with renewable or battery energy. Second, performance improvement by sprinting can provide additional revenue for a cloud provider such as Google, where a large majority of its revenue comes from its data center operations. Conservatively, we assume its revenue is $0.28/KW/min due to operations [14]. However, due to the implementation of renewable energy (PV panels in our design), we estimate the green-power capacity to be $4.74/W [33]. In addition, we ensure that PV panels are amortized over 25 years (lifetime). We assume the cost of batteries to be $50/KW/year [14]. In addition, we include the cost of adding the wax (PCM) into the server cost, although the additional cost is almost negligible representing less than 0.1% of the server cost [21]. Subtracting the capital expenditure of green power and battery, we can then capture the revenue by sprinting as a function of total sprinting durations in a year. As shown in Figure 11, all values to the right of the cross-over point (around 14 hours per year in this case) indicate profitable operations despite of the additional capital cost. This suggests that investing in additional green provision may be worthwhile when conducting as much computational sprinting in data centers as possible.

V. Related Work

Computational Sprinting: Prior work proposed a three-phase methodology that enables safe computational sprinting at data center level which combines data center circuit breakers, energy storage devices and thermal energy storage technologies to control sprinting activities [30]. However, it is no doubt that simply relying on batteries to supply additional power demand cannot fully exploit the potential of sprinting due to strict lifetime constraints. More importantly, the associated carbon footprint expansion still poses significant challenges for large data centers sprinting.

Data center Power provisioning: With the rapid adoption of cloud computing and tremendous data sets flushing into today's data center, the computing resources are increasing continually to their existing sites to support services(i.e.,scaling out), which leads data centers to be power-constrained [10]. That is, today's data centers are already approaching the peak capacity of their power infrastructures. Li et al. proposed a real solar-power server-level prototype called Oasis [10] to scale out data center power capacity economically and sustainably. Our work differs from Oasis in the following three aspects: (1) GreenSprint exploits the sprinting technique to increase the processing capacity in a short period and leverages green power to deal with the power emergency due to the sprinting. Oasis focuses on incremental solar power integration in case of the performance degradation. We emphasizes the role of performance sprinting. (2) GreenSprint pays more attention to the power and performance impact of both core count and frequency scaling techniques. Oasis dynamically adjusts its load processing speed in response to the energy supply only with DVFS. (3) Green energy and battery in GreenSprint are called upon only for occasional power bursts. In contrast, Oasis has more frequent green and battery energy demands because of its scale-out designs.

The high cost of power provisioning and consumption in data centers draws attentions to underprovision the power infrastructure [17], [24], [31], [27], [14], [29], [32]. Different from power capping work, first, data center sprinting with renewable energy can provide more power to servers instead of throttling their power when they need it most. Second, sprinting in green data center is designed for temporarily boosting the computational capacity to achieve better performance, while existing works on data center provisioning are trying to save capital or operating expenses. Also, we must consider the intermittent nature of green power and energy storage device simultaneously. This characteristic of the green mechanism calls for power management schemes to handle the varied power supply and limited energy storage capacity. Green energy as a power source, as opposed to single energy storage supply, has the potential to provide a long-term power supply for data center sprinting.

VI. Conclusion

In this paper, we propose GreenSprint, a renewable energy driven framework that enables data center sprinting to handle occasional workload bursts. We present four strategies to address the challenge imposed by the intermittent and time-varying renewable power supply. We develop an experimental prototype to evaluate our approach. Using representative data center workloads, the results show that our solution can improve the average computing performance significantly by a factor of 4.8x for SPECjbb, 4.1x for Web-Search, and 4.7x for Memcached with sufficient renewable power supply.

Acknowledgment

This work is supported in part by the Wuhan National Laboratory for Optoelectronics Fund under Grant No.0106187015 and No.0106187027, and the US NSF under Grant No.CCF-1704504 and No.CCF-1629625. We would like thank to Rajiv Nishtala for the help on some detailed problems of Faban in last August. We also thank Yaqi Xing and Chuanyi Qi from our lab for the advice on paper writing. We appreciate the suggestions from reviewers to help improve the quality of our paper.

References

[1] Google Summit. http://www.google.com/corporate/datacenter/events/dcsummit2009.html, 2009.
[2] Zh-101 portable electric power fault recorder and analyzer, 2009.
[3] Facebook. Hacking conventional computing infrastructure. http://opencompute.org/, 2011.
[4] Grapesolar. www.grapesolar.com/, 2014.
[5] SPECJBB 2013:Java Business Benchmark. http://www.spec.org/jbb2013/, 2014.
[6] Measurement and instrumentation data center. http://www.nrel.gov/midc/, 2015.
[7] A. Raghavan et al. Computational sprinting. In *HPCA*, 2012.
[8] A. Raghavan et al. Computational sprinting on a hardware/software testbed. In *ASPLOS*, 2013.
[9] A. Raghavan et al. Utilizing dark silicon to save energy with computational sprinting. *Micro*, 2013.
[10] C. Li et al. Enabling datacenter servers to scale out economically and sustainably. In *MICRO*, 2013.
[11] C. Li et al. Enabling distributed generation powered sustainable high-performance data center. In *HPCA*, 2013.
[12] C. Li et al. Power attack defense: Securing battery-backed data centers. In *ISCA*, 2016.
[13] D. Wang et al. Energy storage in datacenters: what, where, and how much? In *SIGMETRICS*, 2012.
[14] D. Wang et al. Underprovisioning Backup Power Infrastructure for Datacenters. In *ASPLOS*, 2014.
[15] G. Wang et al. Increasing large-scale data center capacity by statistical power control. In *EuroSys*, 2016.
[16] H. Esmaeilzadeh et al. Dark silicon and the end of multicore scaling. In *ISCA*, 2011.
[17] L. Barroso et al. The case for energy-proportional computing. *Computer*, 2007.
[18] L. Liu et al. Heb: deploying and managing hybrid energy buffers for improving datacenter efficiency and economy. In *ISCA*, 2015.
[19] M. Ferdman et al. Clearing the clouds: a study of emerging scale-out workloads on modern hardware. In *ASPLOS*, 2012.
[20] M. Haque et al. Greenpar: Scheduling parallel high performance applications in green datacenters. In *ICS*, 2015.
[21] M. Skach et al. Thermal time shifting: leveraging phase change materials to reduce cooling costs in warehouse-scale computers. In *ISCA*, 2015.
[22] R. Nishtala et al. Hipster: Hybrid task manager for latency-critical cloud workloads. In *HPCA*, 2017.
[23] S. Fan et al. The computational sprinting game. In *ASPLOS*, 2016.
[24] S. Govindan et al. Benefits and Limitations of Tapping into Stored Energy for Datacenters. In *ISCA*, 2011.
[25] S. Govindan et al. Leveraging stored energy for handling power emergencies in aggressively provisioned datacenters. In *ASPLOS*, 2012.
[26] T. Keller et al. Ship: A scalable hierarchical power control architecture for large-scale data centers. *TPDS*, 2012.
[27] V. Kontorinis et al. Managing distributed ups energy for effective power capping in data centers. In *ISCA*, 2012.
[28] V. Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 2015.
[29] W. Zheng et al. Exploiting Thermal Energy Storage to Reduce Data Center Capital and Operating Expenses. In *HPCA*, 2014.
[30] W. Zheng et al. Data center sprinting: Enabling computational sprinting at the data center level. In *ICDCS*, 2015.
[31] X. Fan et al. Power provisioning for a warehouse-sized computer. In *ISCA*, 2007.
[32] X. Zhou et al. Underprovisioning the grid power infrastructure for green datacenters. In *ICS*, 2015.
[33] D. Feldman. Photovoltaic (PV) Pricing Trends: Historical, Recent, and Near-Term Projections. *Joint Technical Report*, 2012.