# Online Energy Budgeting for Cost Minimization in Virtualized Data Center

Mohammad A. Islam, Shaolei Ren, A. Hasan Mahmud, and Gang Quan, *Senior Member, IEEE*

**Abstract**—The growing environmental and sustainability concerns have made energy efficiency a pressing issue for data center operation. Governments, as well as various organizations, are urging data centers to cap the increasing energy consumption. Naturally, achieving long term energy capping involves deciding energy usage over a long timescale (without accurately foreseeing the far future) and hence, we call this process "energy budgeting". In this paper, we introduce an online resource management solution, called eBud (energy Budgeting), for a virtualized data center. eBud determines the number of servers, resource allocation to virtual machines and corresponding workload distribution to minimize data center operational cost while satisfying a long term energy cap. We prove that eBud achieves a close-to-minimum cost compared to the optimal offline algorithm with future information, while bounding the potential violation of energy budget constraint, in an almost arbitrarily random environment. We also perform a trace-based simulation study to complement the performance analysis. The simulation results show that eBud reduces the cost by more than $16$ percent (compared to state-of-the-art prediction-based algorithm) while resulting in a zero energy budget deficit. We also perform an experimental study based on RUBiS, demonstrating that in a real life scenario, eBud can achieve energy capping with a negligible increase in operational cost.

**Index Terms**—Computer system organization, resource allocation, virtualization, energy budgeting

## 1 Introduction

The incontestable benefits from resource consolidation have led to a widespread adoption of virtualization in recent years, with $46$ percent penetration among enterprises [1]. The thriving number and scale of virtualized data centers consume a huge amount of energy, surging the operational expenditure through electricity bill. On top of that, data centers are also held accountable for the environmental impacts (e.g., carbon emission) associated with the electricity usage. As a result, nowadays, energy consumption is emerging as a critical concern of data center operation. While economizing short term energy consumption helps to some extent, capping the long term energy has been recently proven to be a more pragmatic move for IT companies [2].

*Why is capping the long term energy consumption important?* According to a recent survey [3], about 77 percent of large data centers[1] are actively pursuing green certifications such as LEED program [4], which requires data centers to restrict the annual energy consumption under a certain level (e.g., reduce by $10-25$ percent, compared to a set of benchmarks). Such certifications also supplement cost savings from reduced energy consumption with tax reductions and favorable accreditations. Moreover, in various regions, governments are imposing tighter energy consumption requirements on large facilities like data centers (e.g. California Energy Efficiency Plan [5]). The U.S. government sets annual energy usage targets for the federal establishments which include the data centers [6]. Industry leaders, such as Google and Microsoft [7], [8], have also been leading by example to achieve "carbon neutrality" (a.k.a., net-zero, completely offsetting electricity usage with renewables), which can also be translated into energy capping if the desired energy consumption cap is interpreted as the total available renewables. Further, energy capping for data centers are motivated by the common business practice of setting a cost budget in advance (e.g., monthly or yearly budget for electricity bill) [9].

*Energy budgeting and its challenges.* Capping long term energy means to meet a certain energy consumption target (energy budget), within a predetermined time frame (budgeting period). This requires deciding the energy usage for a long term, and hence, we term it as energy budgeting. It is different from the well studied problem of power budgeting, which studies the allocation of peak power among servers [10]. Intuitively, the long term energy budget should be allocated in such a way that more energy is consumed when workloads are higher to maintain an acceptable quality of service (QoS). However, the unpredictable nature of far future workloads makes the optimal energy budget distribution particularly challenging. A naive approach can be to distribute the energy budget equally across time, which may lead to unnecessarily high allocation during low workload and/or insufficient energy budget at high workloads. Some existing research attempts to solve the problem [2], [11], [12], but requires accurate prediction of future workloads, which is typically unavailable in practice (due to,

1. "Large" data centers refer to those with more than $5{,}000$ servers each [3].

• *The authors are with Florida International University, Miami, FL 33199. E-mail: {misla012, sren, amahm008, gaquan}@fiu.edu.*

*Manuscript received 1 Mar. 2014; revised 3 Sept. 2014; accepted 10 Nov. 2014. Date of publication 11 Jan. 2015; date of current version 15 June 2016.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TSC.2015.2390231*

for example, unpredicted traffic spikes). On top of the energy budgeting challenge, data center also desires to reduce its operational cost by fine tuning the balance between resource allocation (hence, energy, too) and QoS, which itself is an intriguing problem that has drawn much attention from the research community [10], [13], [14].

*Proposed solution.* Realizing the resource management challenges resulting from the long term energy cap and lack of far future workload information, we propose a provably-efficient online algorithm, called eBud (energy Budgeting), which does not require any long term future information. It determines the number of servers to be powered on, virtual machine (VM) resource allocation and workload distribution among different types of servers to control the data center performance and energy consumption. eBud minimizes the data center operational cost (incorporating both electricity cost and delay cost/penalty) while satisfying the long term energy consumption target. At the core of our solution is the energy budget deficit queue (described in Section 3) which tracks how far the energy consumption is from the long term target. The energy budget deficit queue steers the resource management decisions towards meeting the long term energy consumption target: the longer queue, the more focus on reducing energy consumption. The queue embedded in eBud does not adversely affect the data center QoS or electricity cost significantly and enables eBud to deliver a close-to-optimal operational cost.

We perform a trace-based simulation to evaluate eBud and compare it to the state-of-the-art prediction-based method. Our simulation results show that eBud can reduce the average cost by over 16 percent while satisfying the long term energy budget constraint. Sensitivity studies also demonstrate that eBud is robust against inaccurate workload prediction as well as consideration of server switching cost. To support our simulation studies, we present results of a scaled down experiment with RUBiS benchmark application and show that eBud can meet a long term energy consumption target in real life scenario while incurring a small increase in average cost.

The rest of this paper is organized as follows. Section 2 describes the model. In Section 3, we present the problem formulation and develop our online algorithm, eBud. Sections 4 and 5 provide simulation and experimental results to support our analysis. Related work is reviewed in Section 6 and finally, concluding remarks are offered in Section 7.

## 2 Model

A discrete time model, which equally divides the total budgeting period into $K$ equal time slots indexed by $t = 0,\ 1, \ldots, K-1$, is considered in this paper. The resource management decisions, governed by eBud to meet the long term energy budget, are updated periodically at the beginning of each slot. The granularity of the time slots are limited by the minimum time gap between resource management updates. In the following sections, we present the modeling details for the data center, server power and workload.

### 2.1 List of Notations

| | |
|---|---|
| $I$ | Types of physical servers. |
| $J$ | Types of workloads. |
| $M_i$ | Maximum number of type-$i$ servers. |
| $m_i$ | Number of type-$i$ servers turned on. |
| $\mathrm{VM}_{i,j}$ | VM at type-$i$ server, serving type-$j$ workloads. |
| $x_{i,j}$ | CPU resource of $\mathrm{VM}_{i,j}$. |
| $\lambda_{i,j}$ | Workload of $\mathrm{VM}_{i,j}$. |
| $\mu_{i,j}$ | Service rate of $\mathrm{VM}_{i,j}$. |
| $s_i$ | CPU resource of type-$i$ servers. |
| $p_i$ | Power consumption of type-$i$ servers. |
| $u_i$ | CPU utilization of type-$i$ servers. |
| $w$ | Electricity price. |
| $e$ | Electricity cost. |
| $g$ | Total cost. |
| $V$ | Cost-capping parameter. |
| $q$ | Budget deficit queue length. |

### 2.2 Data Center

We lay down the foundation of our model by introducing the data center architecture considered in this paper. We consider a virtualized data center where the VMs are hosted in $I$ different types of physical servers distinguished by their power consumption and processing capacity. Throughout the paper, servers and physical servers are used interchangeably wherever applicable. Each server hosts a set of VMs and contains a VM monitor (VMM, also called hypervisor) that is responsible for allocating hardware resources to the hosted VMs. The data center serves $J$ different types of workloads through VMs hosted in different servers. The number of VMs to serve each workload type is determined by the resource management algorithm. Each server hosts up to $J$ VMs, each corresponding to one type of workload, and each VM is assigned a fraction of the total available CPU resource of that server. We denote the VM serving type-$j$ workload at type-$i$ server by $\mathrm{VM}_{i,j}$ and its CPU resource allocation by $0 \leq x_{i,j} \leq s_i$, where $s_i$ is the CPU resource available at type-$i$ servers (determined by resource management algorithm and measured in, e.g., GHz excluding the CPU consumption by VMM) for processing VM workloads. At any time slot $t$, the CPU resource allocation $x_{i,j}$ is considered identical among the same type of servers, with the following rationale. For homogeneous servers (i.e., the same type of servers in our study), equally distributing workloads across active servers is optimal in terms of minimizing the cost, and hence with everything being the same (i.e., equal workload, equal physical configuration), it is natural to allocate the same CPU resource to VM across all active servers of the same type.

### 2.3 Server Power

As CPU power is the dominant component of server total power consumption (typically $40-50$ percent) [15], we mainly focus on CPU allocation for VMs while considering other resources (e.g., memory, disk) as sufficient and non-bottleneck resources that consume a power with relatively less variation. Although this assumption may not hold for all application scenarios (e.g., memory/disk power consumption may vary considerably for I/O-intensive workloads), we note that it is reasonably accurate for CPU-intensive workloads that are the main concentration of our study [16],

[17] and that the assumption will also be validated using real life experiments running workloads that are not purely CPU-intensive. We express the average power consumption[2] and peak power of server type $i$ as [16], [18]

$$p_i(u_i) = \left[p_{i,s} + u_i \cdot p_{i,c}\right] \cdot \mathbf{1}_{(u_i>0)}, \tag{1}$$

$$\hat{p}_i = p_{i,s} + p_{i,c}, \tag{2}$$

where $u_i$ is CPU utilization of type-$i$ server (that will be specified in the next section), $p_{i,s}$ is the static power regardless of the workloads as long as the server is turned on, and it also captures the power consumption resulted from the CPU usage of the VMM (responsible for managing the VMs in a server), $p_{i,c}$ is the computing power incurred when the server is operating, and the indicator function $\mathbf{1}_{(u_i>0)} = 1$ if and only if server utilization $u_i > 0$. Thus, considering $m_i(t) \le M_i$ as the number of type-$i$ servers turned on, the servers' total power consumption during time $t$ is given by

$$p(\mathbf{u}(t), \mathbf{m}(t)) = \sum_{i=1}^{I} m_i \cdot p_i(u_i(t)), \tag{3}$$

where $\mathbf{u}(t) = (u_1(t), u_2(t), \ldots, u_M(t))$ is the vector of average server utilization, $\mathbf{m}(t) = (m_1(t), m_2(t), \ldots m_M(t))$ is the vector of number of turned on servers of each type. Note that we isolate the non-IT power consumption (e.g., by cooling) which is in proportion to the servers' power.

### 2.4 Workload

We focus on delay-sensitive interactive workloads like web applications (as in [14], [16]), whereas delay-tolerant batch workloads (e.g. scientific calculations) can be captured separately through a batch job queue as considered by several existing studies [16]. Arrival rate of the $J$ different types of workloads are denoted by $a_j(t) \in [0, a_{j,\max}]$, where $j = \{1, \ldots, J\}$. As assumed in prior work [2], [14], [19], the value of $a_j(t)$ is accurately available at the beginning of each time slot $t$, as can be achieved by using various techniques such as regression analysis, while inaccurate knowledge of the service rate/workload arrival rate will be investigated in the simulation section.

Each type of workload is distributed among VMs placed in different types of servers. Let $\lambda_{i,j}$ be the type $j$ workload processed at type-$i$ server. Thus, the workload of $\mathrm{VM}_{i,j}$ is $\frac{\lambda_{i,j}(t)}{m_i(t)}$, because servers of the same type get equal workload distributions. The service rate (i.e., how many jobs can be processed on average in a unit time) of $\mathrm{VM}_{i,j}$ is given by $\mu_{i,j} = \mu_{i,j}(x_{i,j,})$. The function $\mu_{i,j}(\cdot)$ maps the allocated CPU resource to the service rate. While in general it is non-trivial to accurately obtain the mapping $\mu_{i,j}(\cdot)$ [20], [21], we note that the service rate of CPU-intensive jobs can be approximated as an affine function of the allocated CPU resource (provided that memory, disk, etc are non-bottleneck) [17]. Thus, as in [16], [18], we assume in the following analysis that $\mu_{i,j}(\cdot)$ is exogenously determined, for $i = 1, 2, \ldots, I$ and $j = 1, 2, \ldots, J$, while noting that modeling the function $\mu_{i,j}(\cdot)$ can be done using the techniques developed in [20], [21], that are beyond the scope of our study.

2. This is directly proportional to energy consumption, and since the length of each time slot is the same, we use (1) as the measure of each server's energy consumption in a time slot.

Now, we are ready to derive the average server utilization $u_i(t)$ as follows:

$$u_i(t) = \sum_{j=1}^{J} \frac{\lambda_{i,j}(t)/m_i}{\mu_{i,j}(t)} \cdot \frac{x_{i,j}(t)}{s_i(t)} \tag{4}$$

where $\frac{\lambda_{i,j}(t)/m_i}{\mu_{i,j}(t)}$ is the utilization of $\mathrm{VM}_{i,j}$ and $\frac{x_{i,j}(t)}{s_i(t)}$ is portion of server type $i$'s CPU resources allocated to $\mathrm{VM}_{i,j}$. This completes the expression of average server power consumption[3] in (1).

## 3 Online Energy Budgeting

In this section, we specify the data center costs and the optimization problem formulation for the energy budgeting problem. Then, we develop our online energy budgeting algorithm, eBud, which makes resource management decisions without far future information. We conduct a performance analysis and show that eBud can perform close to the optimal offline algorithm while satisfying the long term energy budgeting constraint.

### 3.1 Data Center Costs

We focus on minimizing operational cost of the data center rather than capital cost (e.g., building data centers). While electricity cost takes up a significant portion of the data center operational cost, we also incorporate the delay cost (specified in the following section) which quantifies the data center performance and affects the user experiences as well revenues [19].

*Electricity cost.* We denote the electricity price at time $t$ by $w(t)$, which is known to the data center operator no later than the beginning of each time slot but may change over time if the data center participates in a real-time electricity market [13], [14]. Thus, the total electricity cost at time $t$ can be expressed as

$$e(\mathbf{u}(t), \mathbf{m}(t)) = w(t) \cdot p(\mathbf{u}(t)). \tag{5}$$

While we use a linear function (5) to model the electricity cost, our analysis is applicable for nonlinear (i.e convex) cost functions where the data center is charged at higher rate as their power consumption increases. Although considering on-site renewable energy and/or energy storage devices such as batteries can further reduce the electricity cost [12], we do not consider them in our work as they are complimentary to our study and our main focus is on deciding resource management and workload distribution in an online fashion while satisfying the energy budget constraint.

*Delay cost.* As in prior research [14], [19], we denote the delay cost at $\mathrm{VM}_{i,j}$ by a convex function $d_{i,j}(\lambda_{i,j}, \mu_{i,j}, m_i)$, which is intuitively increasing in $\lambda_{i,j}$ and decreasing in $\mu_{i,j}$ and $m_i$. To facilitate our analysis, we model the service process at each VM as an M/G/1/PS queue and use the average response time (multiplied by the arrival rate) to represent the delay cost. Specifically, it is well known that the average response time for the M/G/1/PS is $\frac{1}{\mu_{i,j} - \frac{\lambda_{i,j}}{m_i}}$ [22] and hence, the total delay cost at time $t$ can be written as

$$d(\boldsymbol{\lambda}(t), \boldsymbol{\mu}(t), \mathbf{m}(t)) = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{m_i \cdot \lambda_{i,j}(t)}{m_i \cdot \mu_{i,j}(t) - \lambda_{i,j}(t)}, \tag{6}$$

3. The domain0/root VM power consumption is absorbed in the "static" power $p_{s,i}$ of server $i$.

in which $\boldsymbol{\lambda}(t) = (\lambda_{1,1}(t), \ldots, \lambda_{I,J}(t))$, $\boldsymbol{\mu}(t) = (\mu_{1,1}(t), \ldots, \mu_{I,J}(t))$, $\mathbf{m}(t) = (m_1(t), m_2(t), \ldots m_I(t))$, and we ignore the network delay cost that can be approximately modeled as a certain constant [14] and added into (6) without affecting our approach of analysis. The M/G/1/PS model has been widely used to get a reasonable approximation of the service process, and our study does not preclude the incorporation of other delay performance metrics (e.g., 95 percent delay), although the analytic convenience may be forfeited. Finally, we note that the delay cost model in (6) implicitly assumes that VMs are perfectly isolated without interfering with each other. While this may not be true in heavy traffic scenarios (e.g., due to cache, I/O contention) as pointed out by the existing research [20], we consider perfect isolation and use the delay cost model only as an approximate indication for the actual delay performance (as studied in [16]). In other words, the model is only intended to facilitate the design of our online algorithm, while the actual decision and delay performance should still be appropriately calibrated online to account for the factors (e.g., VM interference, imperfect workload monitoring) that are not captured by our model. In our experimental study, the assumption will be further validated using a real life system.

*Operational costs.* We construct the data center operational cost by combining the electricity and delay cost in a parameterized cost function as follows:

$$g(\mathbf{m}(t), \mathbf{x}(t), \boldsymbol{\lambda}(t)) = e(\mathbf{u}(t), \mathbf{m}(t)) + \beta \cdot d(\boldsymbol{\lambda}(t), \boldsymbol{\mu}(t), \mathbf{m}(t)), \tag{7}$$

where the server utilization vector $\mathbf{u}(t)$ and VM service rate matrix $\boldsymbol{\mu}(t)$ are functions of the VM resource allocation $\mathbf{x}(\mathbf{t})$, and $\beta \geq 0$ is weighting parameter for delay cost relative to the electricity cost [14], [19]. By tuning $\beta$, data center operator can decide a balance between electricity bill and QoS.

## 3.2 Problem Formulation

The optimization problem that minimizes the virtualized data center's operational cost subject to a long term energy capping is presented below:

$$\mathbf{P1}: \quad \text{minimize} \quad \bar{g} = \frac{1}{K} \sum_{t=0}^{K-1} g(\mathbf{m}(t), \mathbf{x}(t), \boldsymbol{\lambda}(t)), \tag{8}$$

$$\text{subject to} \sum_{t=0}^{K-1} p(\mathbf{u}(t), \mathbf{m}(t)) \leq Z. \tag{9}$$

The optimization objective (8) in the above problem formulation indicates the data center aims at minimizing its long term average operational cost where $K$ is the total number of time slots in the budgeting period. The long term energy capping is specified by (9), where $Z$ is target total energy consumption during the budgeting period. We note that setting the energy cap $Z$ is out of the scope of our paper but non-trivial. One method used by LEED program for an existing data center is to set $Z$ based on the data center's historical energy consumption or energy consumption by similar buildings [4].

The optimization problem is also subject to some additional constraints which arise from the choice of control variables (number of servers to be powered on, VM resource allocations and workloads distribution) in the optimization. The constraints are listed below:

$$\sum_{i=1}^{I} m_i(t) \cdot \hat{p}_i(x_i(t)) \leq \hat{P}, \tag{10}$$

$$0 \leq m_i(t) \leq M_i, \forall i, \tag{11}$$

$$\sum_{j=1}^{J} x_{i,j}(t) \leq s_i(t), \forall i, \tag{12}$$

$$0 \leq \lambda_{i,j}(t) \leq \theta \cdot m_i(t) \cdot \mu_{i,j}(t), \forall i, \forall j, \tag{13}$$

$$\sum_{i=1}^{I} \lambda_{i,j}(t) = a_j(t), \forall j. \tag{14}$$

Constraints (10) and (11) limit data center peak power and the maximum number of servers that can be brought into service, where $\hat{P}$ is the data center peak power and $M_i$ is the maximum number of $i$-type servers. Constraint (12) bounds the VM CPU allocation within the available server resource. Constraints (13) and (14) specify the conditions for workload distribution and no workload dropping, respectively, where $\theta \in (0, 1)$ specifies the maximum utilization of a single VM.

While the optimal solution to **P1** provides the resource allocations for the minimum achievable average cost, constraint (9) couples together the resource management decisions of all the time slots, and hence requires the complete offline information (i.e., workload arrivals and electricity prices) over the entire budgeting period. In practice, however, it is very difficult to predict the workload in advance for longer periods because of the traffic spikes caused by breaking events [23], [24]. Moreover, **P1** has an integer decision variable (i.e., number of servers (11)), turning it into a mixed-integer nonlinear programming problem spanning over all the time slots and making **P1** computationally exhaustive even with the long term future information accurately known a priori. Next, we propose an efficient online algorithm to address these challenges.

## 3.3 Algorithm for Online Energy Budgeting (eBud)

Developed based on Lyapunov optimization technique [25], eBud is purely online which requires only the currently available information yet performs close to the optimal offline solution in terms of cost minimization while satisfying the long term energy budget. eBud also gives data center operator the flexibility of run-time adjustment of trade-off between cost minimization and the budget constraint. Next, we describe eBud in detail.

*Budget deficit queue.* To enable online resource management by removing the resource management decision coupling in **P1**, we integrate the long term budgeting constraint with the optimization objective function by introducing a virtual budget deficit queue. The queue tracks the deviation from the long term target, and assuming $q(0) = 0$, its dynamics evolves as follows:

$$q(t+1) = \left[ q(t) + p(\mathbf{u}(t), \mathbf{m}(t)) - \frac{Z}{K} \right]^{+}. \tag{16}$$

**Algorithm 1.** eBud

```
1: Input $\mathbf{a}(t)$ and $w(t)$ at the beginning of each time $t = 0, 1, \ldots, K-1$
2: if $t = rT, \forall r = 0, 1, \ldots, R-1$ then
3:     $q(t) \leftarrow 0$ and $V \leftarrow V_r$
4: end if
5: Choose $\mathbf{m}(t), \mathbf{x}(t)$ and $\boldsymbol{\lambda}(t)$ subject to (10), (11), (12), (13) and (14) to minimize

   $\mathbf{P2}: \quad V \cdot g(\mathbf{m}(t), \mathbf{x}(t), \boldsymbol{\lambda}(t)) + q(t) \cdot p(\mathbf{u}(t), \mathbf{m}(t))$   (15)

6: Update $q(t)$ according to (16).
```

A positive queue length at any time indicates that the data center has drawn more electricity energy than the allocated budget thus far and needs to consume less energy in the consecutive time slots to offset the excess use. Incorporating the energy budget deficit queue as a guidance, we develop our online algorithm, eBud, as presented in Algorithm 1.

eBud is an online algorithms which only requires the currently available information (i.e., $a(t)$, $w(t)$) as inputs, while its output is: how many servers of each type are turned on, CPU resource allocation to VMs in each server, and workload distribution among the servers turned on. In Algorithm 1, we use $V_1, V_2, \ldots, V_R$ to denote a sequence of positive control parameters (also referred to as cost-capping parameters) which determine the relative weight of meeting long term budget over cost minimization. Lines 2-4 reset the energy budget deficit queue at the beginning of each frame $r$, such that the cost-capping parameter $V$ can be adjusted and the energy budget deficit in a new time frame will not be affected by its value resulting from the previous time frame. Line 5 solves a one-time mixed integer optimization problem to determine the resource management for the corresponding time slot. At the end of each time slot, the energy budget deficit queue is updated for usage in next time slot.

Finally, it is worth mentioning that **P2** is still a mixed-integer problem that may be solved using software packages or other techniques (e.g., branch and bound [26]). However, although the complexity of **P2** is exponential in the number of server types, eBud is practically realizable, because the resource management decision is only made once every time slot (i.e., the total intolerable complexity is distributed among all time slots). Note further that to achieve energy capping, the delay performance needs to be compromised (but still subject to a performance constraint), as demonstrated in other energy-saving literature [27]. Indeed, Facebook is intentionally hitting the performance constraint to save energy by dynamically providing computing resources, rather than leaving all the servers on to achieve the best possible delay performance [28].

### 3.4 Performance Analysis

This section presents the performance analysis of eBud in Theorem 1, whose proof follows sample-path Lyapunov technique in [25] and details are omitted for brevity with a sketch provided in the Appendix.

**Theorem 1.** *For any $T \in \mathbb{Z}^+$ and $R \in \mathbb{Z}^+$ such that $K = RT$, the following statements hold.*

***a.*** *The energy capping constraint is approximately satisfied with a bounded deviation:*

$$\frac{1}{K}\sum_{t=0}^{K-1} p(\mathbf{u}(t), \mathbf{m}(t)) \leq \frac{Z}{K} + \frac{\sum_{r=0}^{R-1}\sqrt{C(T) + V_r(G_r^* - g_{\min})}}{R\sqrt{T}}, \tag{17}$$

*where $C(T) = B + D(T-1)$ with $B$ and $D$ being certain finite constants, $G_r^*$ is the minimum average cost achieved over the r-th frame by the optimal offline algorithm with $T$-slot lookahead information (specified in the appendix), for $r = 0, 1, \ldots, R-1$, and $g_{\min}$ is the minimum per-slot cost that can be achieved by any feasible decisions throughout the budgeting period.*

***b.*** *The average cost $\bar{g}^*$ achieved by eBud satisfies:*

$$\bar{g}^* \leq \frac{1}{R}\sum_{r=0}^{R-1} G_r^* + \frac{C(T)}{R} \cdot \sum_{r=0}^{R-1}\frac{1}{V_r}. \tag{18}$$

Theorem 1 shows that the cost achieved by eBud is within an additive penalty of $O(1/V)$ compared to the minimum cost achieved by the offline algorithm with $T$-step lookahead (specified in the appendix), while the energy capping constraint is guaranteed to be approximately satisfied with a bounded deviation. When $V$ increases, eBud is closer to the offline algorithm, whereas the deviation from the capping constraint may be larger in the worst case, and vice versa.

We note that despite the provably-efficient performance, eBud based on Lyapunov optimization has a shortcoming: slow convergence rate. That is, eBud usually requires quite a few time slots before the algorithm can yield a stable performance (both in terms of cost and in terms of energy capping).

### 3.5 Updating $V$ Dynamically

As shown in Theorem 1, the value of $V$ is very important in eBud. $V$ determines how close the average cost achieved by eBud will be to that of the optimal offline algorithm, which is the theoretical lower bound on average cost. It also sets the limit on the maximum deviation from the long term energy capping target. A large value of $V$ will reduce the average cost performance gap between eBud and optimal offline, while the deviation from energy capping target will be higher. While the appropriate value of $V$ is not possible to accurately determined at the beginning of the entire budgeting period, $V$ can be updated at run time to cope with the operation need. Starting with any initial value of $V$, after some time if the data center operator sees that the energy consumption is exceeding the set target so far, it can reduce $V$ so that the algorithm emphasizes energy capping over cost reduction, and vice versa. Based on this intuition, we implement the following to update the $V$ periodically,

$$V_{\tau+1} = \max\{V_\tau + \alpha \cdot z_\tau,\ V_{\min}\}, \tag{19}$$

where $V_\tau$ is the value of $V$ after the $\tau$-th update, $\alpha > 0$ is a scaling parameter, $V_{\min} > 0$ is the smallest possible value of $V$ to avoid zero value, and $z_\tau = \frac{Z}{K} - \frac{1}{t_{\tau+1}}\sum_{t=0}^{t_{\tau+1}-1} p(\mathbf{u}(t), \mathbf{m}(t))$ is the difference between the average allocated budget and the energy consumption up to the time slot $t_{\tau+1}$ during

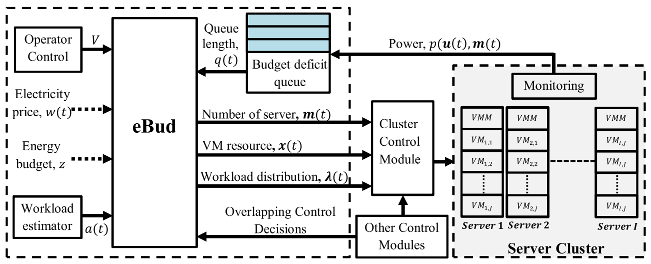

Operator Control
V
Electricity price, $w(t)$
Energy budget, $z$
Workload estimator
$a(t)$
eBud
Queue length, $q(t)$
Budget deficit queue
Power, $p(\mathbf{u}(t), \mathbf{m}(t))$
Monitoring
Number of server, $\mathbf{m}(t)$
VM resource, $\mathbf{x}(t)$
Workload distribution, $\boldsymbol{\lambda}(t)$
Cluster Control Module
Overlapping Control Decisions
Other Control Modules
VMM
Server 1
Server 2
Server I
Server Cluster

Fig. 1. System block diagram.

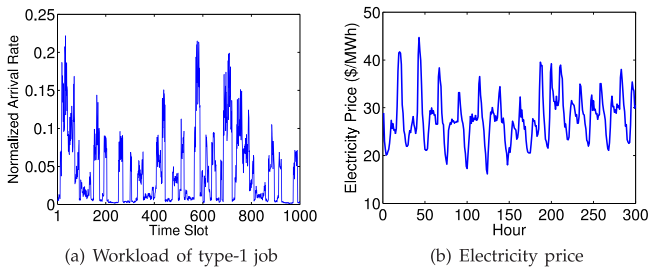


Fig. 2. Workload trace and electricity price.

which $V$ is updated for the $(\tau + 1)$-th time. Using the proposed method, eBud can be applied based on online information and, with an initial input of $V$ which does not need to be accurate, will automatically guide itself towards energy capping constraint satisfaction. We further investigate the impact of $V$ on eBud in the simulation and experiment.

### 3.6 Integration of eBud with Existing Systems

Before concluding this section, we briefly discuss how eBud can be seamlessly integrated with the existing resource management module in data centers. As illustrated in Fig. 1, eBud is placed as part of the data center resource management module. The control decisions made by eBud are sent to the server cluster control module, which then directly controls the servers and VMs. The virtual budget deficit queue, which is a part of eBud, collects the energy usage data from the data center monitoring system. While eBud decides the number of servers as well as VM CPU resource allocation, supplementary control modules (i.e., cooling system control, non-CPU VM resource allocation, network system control) can be appended after eBud to improve the performance. Any existing control module that makes overlapping decisions is placed before eBud, and can be adopted as additional constraints in our algorithm.

## 4 Simulation

This section presents trace-based simulation studies of a data center to validate our analysis and evaluate the performance of eBud. We first present our data sets and then show the simulation results.

### 4.1 Data Sets

We consider a data center with 40,000 physical servers. There are four types of servers with 10,000 of each type. The normalized server service rate and power consumption of the server types are: (2.7, 250 W), (2.8, 260 W), (2.9, 270 W), and (3.0, 275 W). The duration of each time slot is 10 minutes. The budgeting period in our study is one month, and the default total energy budget is set to 6,009 MWh, which is 85 percent of total energy consumption if there were no long term budget constraint achieved by the capping-unaware cost saving algorithm (specified later). The weighting parameter converting the delay to monetary cost is $\beta = 0.002$. The parameter $\theta$ that determines maximum VM utilization is set to 95 percent.

- *Workloads.* We consider that the data center serves four different types of workloads. The workload traces for our simulations are taken from the research publication [27] and originally profiled from I/O traces taken from 6 RAID volumes at Microsoft Research (MSR) Cambridge. The traced period was 1 week starting from 5PM (GMT) on February 22, 2007 [27]. We first repeat the trace after adding 30 percent random noise to extend it to one month, and then add 50 percent random variation to generate the workload of the four types of jobs. Fig. 2a illustrates the workload for type-1 jobs, where the workloads are normalized with respect to the total maximum service capacity of the data center.
- *Electricity price.* As in [13], [14], we consider that the data center participates in a real-time electricity market and obtain from [29] the hourly electricity price for Mountain View, California, during January, 2012.
- *Others.* We consider that the servers can host VMs to serve any type of workload. The VM service rate (i.e., maximum number of jobs served per unit time) is proportional to the allocated CPU resource, while the proportionality factor depends on the workload type as well as the server type: in our simulation, we use 16 different constants of proportionality to simulate four different types of jobs served by four different types of server in the data center. In particular, the constants of proportionality are pre-determined before the simulation but randomly chosen from the following set: $[0.80, 0.85, 0.90, 0.95, 1.00, 1.05, 1.10, 1.15, 1.20, 1.25]$.

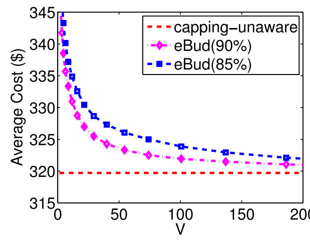
Since the access to commercial data centers is unavailable, we obtain the trace data from sources available to us, but it captures the variation of workloads and electricity prices over the budgeting period.

### 4.2 Simulation Results

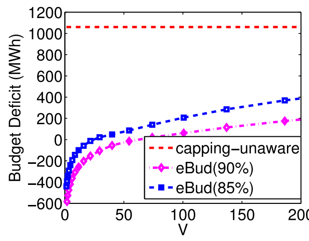
We now present the simulation results based on the above settings. We first study the execution of eBud, followed by performance comparison of eBud with the best known existing solution. We then show the impact of energy budget $Z$, and finally present a sensitivity study showing the robustness of eBud.

#### 4.2.1 *Execution of eBud*

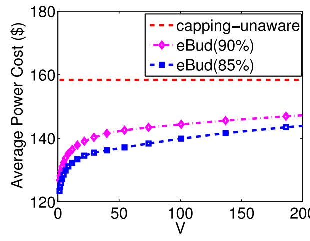
We first consider cases where $V$ is kept constant throughout the budgeting period. Figs. 3a and 3b show the impact of $V$ on the average cost per time slot (i.e., $\bar{g}$) and the total energy budget deficit (i.e., budget deficit at the end of the budgeting period). The negative budget deficit in Fig. 3b indicates that the total energy usage is less than the budget



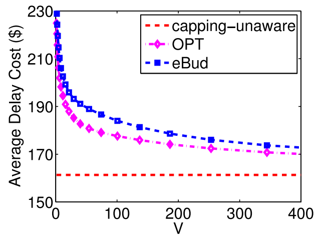
(a) Average cost approaches lower bound as $V$ increases.



(b) Zero budget deficit is achieved by eBud at different $V$ for different budget.

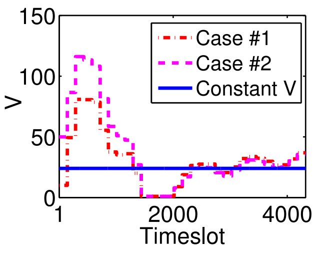


(c) Electricity cost versus $V$.

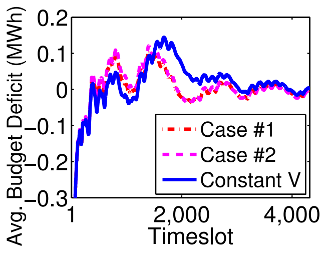


(d) Delay cost versus $V$.

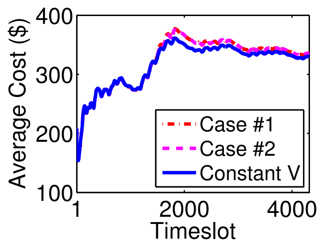
Fig. 3. Impact of $V$.



(a) Dynamic $V$

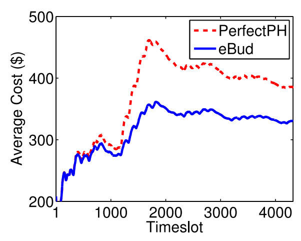


(b) Average budget deficit.

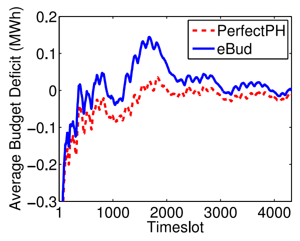


(c) Average cost.

Fig. 4. Dynamically updating $V$ every day. Case #1: initial $V = 10$. Case 2: initial $V = 50$. Constant $V = 24$. $\alpha = 50$.



(a) Average cost.



(b) Average budget deficit.

Fig. 5. Comparison with prediction-based methods.

amount. We also capture the effect of allocated energy budget using two different budget levels, at 85 percent (the default value) and 90 percent of total energy consumption without any energy cap. The results conform with our analysis of dependence of eBud performance on the value of $V$. We see that the average cost decreases with increase in $V$, while the budget deficit increases. This is because when the value of $V$ is large, average cost minimization has a greater weight than energy capping. If a very large $V$ (e.g., infinity) is used, relative weight of the energy capping becomes almost zero, thus making eBud act like an algorithm that only minimizes average cost while disregarding energy capping. The average cost achieved in such a case is the lower bound on average cost achievable by any algorithm with an energy cap. We also see from Fig. 3b that zero budget deficit is achieved by eBud at lower $V$ for lower budget: at $V = 65$ for 90 percent budget and at $V = 25$ for 85 percent budget. The average cost at the zero budget deficit is fairly close to the lower bound, greater only by 1 and 3.1 percent at 90 and 85 percent budget, respectively. This shows the effectiveness of using eBud to cap long term energy without much increase in the data center cost.

In Fig. 4, we show the effectiveness of using (19) to dynamically update $V$. We show two cases with different starting values of $V$ and compare them with the average budget deficit and average cost of using constant $V$. Fig. 4a shows the evolution of $V$ over time. We see that even though the starting $V$ of case #2 is five times that of case #2, the pattern of dynamic $V$ over the entire budgeting period is similar, with larger variations at the beginning because of different initial values; beyond time slot 1,500, they become almost identical. We see in Fig. 4b that despite starting with different $V$, in both cases, the long term energy capping is satisfied. We also see in Fig. 4c that the average cost is very close to that of the constant $V$, with only 1.6 percent higher for both cases. Thus using (19), the data center operator can be relieved from finding the appropriate $V$ at the beginning of a budgeting period.

### 4.2.2 *Comparison with Prediction-Based Methods*

This simulation compares eBud with the best known existing solution—prediction-based method for energy/cost capping as considered in [2], [11], [12]. We incorporate the nonlinear delay function to the existing prediction-based methods and consider a heuristic variation as follows.

• Perfect Prediction Heuristic (*PerfectPH*): The data center operator leverages perfect prediction of 10-minute workload arrival rates for the next 48 hours and allocates the energy budget in proportion to the workloads. When operating online, the operator minimizes the cost subject to the allocated 10-minute energy budget; if no feasible solution exists for a particular time slot (e.g., workload spikes), the operator will minimize the cost without considering the allocated energy budget.

Fig. 5 shows the comparison between eBud and the prediction-based PerfectPH in terms of the average cost and budget deficit per time slot.[4] Fig. 5a demonstrates that eBud is more cost-effective compared to the prediction-based methods with a cost saving of more than 16 percent over the budgeting period of one month, while both algorithms meet the long term budget constraint. The higher cost of PerfectPH is the result of short term prediction

4. The average at time $t$ in Fig. 5 is obtained by summing up all the values from time 0 to time $t$ and then dividing the sum by $t + 1$.

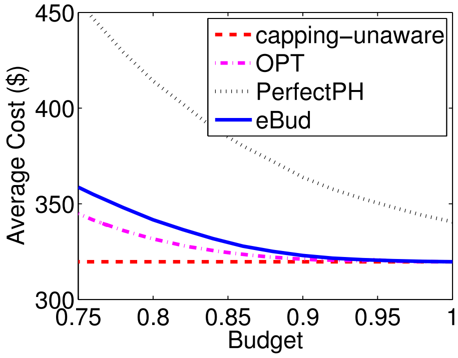
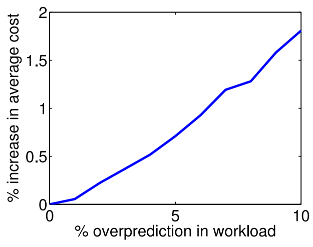


Fig. 6. Impact of energy budget.

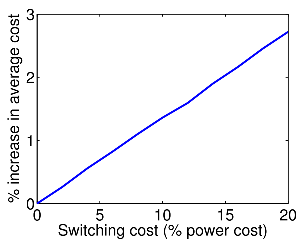
while allocating the energy budget. Without foreseeing complete offline information, PerfectPH may over-allocate the energy budget at inappropriate time slots, and consequently have to set a stringent budget for certain time slots when the workload is high. To satisfy the stringent energy budget of these time slots, PerfectPH reduces server speed and/or turns off servers, resulting in increased delay cost and hence a high average cost. By contrast, eBud does not impose any per-slot hard energy budget constraint, and can focus on cost minimization during workload spikes with the energy budget temporarily violated. In Fig. 5b, the average budget deficit is shown over the budgeting period. We see that PerfectPH has a negative budget deficit, because for some time slots, PerfectPH assigns more budget than the optimum energy consumption. In other words, the increase in electricity cost for using up the whole energy budget is greater than the reduction in delay cost. Thus, there are unused energy budgets in these time slots that result in the negative budget deficit. Note that, if a longer-term prediction is combined, eBud can naturally further reduce the cost, but the cost saving potential is quite limited, because Fig. 3a already demonstrates that eBud is fairly close to the lower bound on the cost while satisfying the budget constraint. This implies that only using 10-minute-ahead prediction in eBud is sufficiently good in terms of cost minimization.

#### 4.2.3 *Impact of Long Term Energy Budget $Z$*

We now show in Fig. 6 the impact of long term energy budget $Z$ on the cost. Under our simulation settings, the capping-unaware algorithm consumes $7{,}069$ MWh electricity energy over one month, which we normalize to $1$. We vary the energy budget from $0.75$ to $1$ and observe the average cost achieved by eBud. We appropriately choose $V$ such that eBud achieves a zero budget deficit. We see that average cost of eBud increases as the energy budget is shrinking. This shows the performance penalty (in terms of average cost) associated with meeting a long term energy target, and naturally a lower budget results in higher performance penalty. However, we see that given a $75$ percent energy budget, average cost of eBud only exceeds the that of capping-unaware algorithm by approximately $12.2$ percent. We also show the impact of energy budget on the optimal offline algorithm, called OPT, and PerfectPH. The average cost gap between eBud and OPT, increasing with decreasing budget, remains very small, only $4$ percent at $75$ percent budget.



(a) Workload prediction error



(b) Incorporate server switching cost.

Fig. 7. Robustness against workload prediction error and switching cost.

This supports our analytical claim that eBud performs close to optimal offline algorithm. PerfectPH, on the other hand, incurs the greatest cost exceeding the capping-unaware algorithm by $6.5$ percent at even $100$ percent budget and $43$ percent at a $75$ percent budget. These results indicate the effectiveness of eBud in capping long term energy consumption at different budget levels.

#### 4.2.4 *Sensitivity Study*

Now, we investigate the robustness of eBud in the following two scenarios.

*Workload overestimation.* Since in practice, it may not be always possible to accurately predict the workload arrivals, a conservative approach can be overestimating the workload, and as a result, keep more server turned on to cope with unexpected traffic spikes. The overestimation can also be considered equivalent to imperfect modeling of VM service rate. In Fig. 7a, we show the percentage increase in average cost caused by overestimated workload for eBud. The cost-carbon parameter $V$ is appropriately adjusted to achieve zero budget deficit. We see that the average cost for eBud increases less than $2$ percent when we overestimate the workload by $10$ percent. This indicates that eBud is robust against workload prediction error.

*Considering server switching cost.* We also study the performance of eBud when the server toggling power[5] is added to the power cost. We model the switching cost (both "Power up" and "Shutdown") of a server as percentage of the server's peak energy consumption and calculate the total switching power consumption by multiplying the switching power by the change in number of servers between time slots. In Fig. 7b, we show that the average cost increases less than $3$ percent even when we consider a switching cost of $20$ percent. This shows that the performance of eBud is not degraded when the server switching cost is considered.

## 5 Experiment

In this section, we present our experiment results to show the effectiveness of eBud in satisfying long term energy budget constraint in a scaled down prototype platform. We first describe our experiment setup and then present the experiment results.

### 5.1 Experiment Setup

There are two virtualized servers and one client machine in our experiment. Fig. 8 shows the connectivity among

5. Power consumption during the "power up" or "shutdown" period when server cannot serve workload but consumes power.

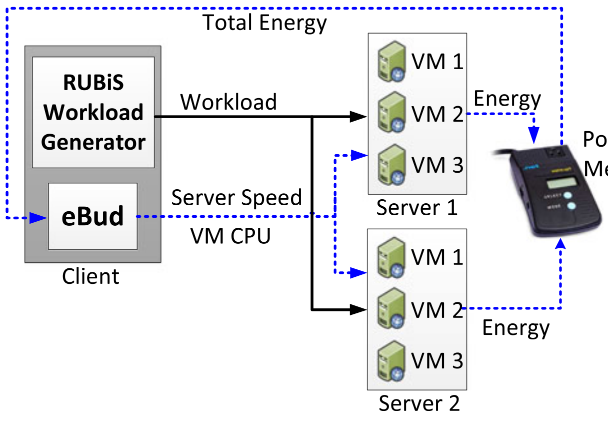


Fig. 8. Experiment setup for validating eBud.

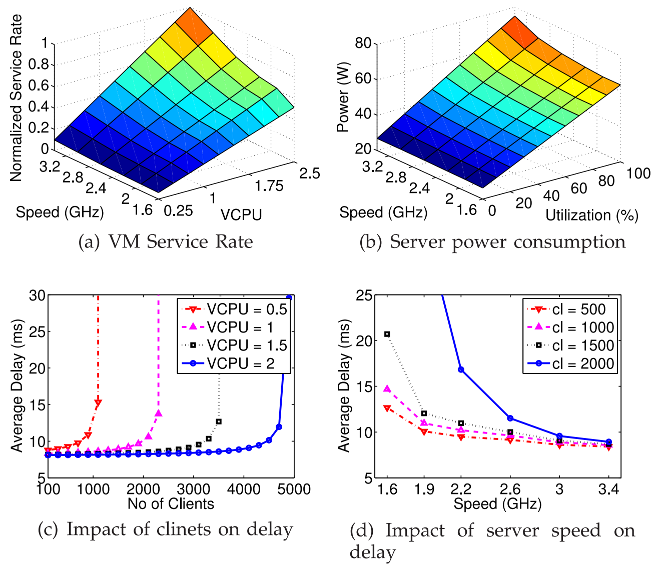


Fig. 9. Server and VM model for experiment.

different components of the experiment. The resource (e.g., server CPU speed) management algorithm eBud and RUBiS workload generator are placed in the client machine, while VMs in the two servers host the RUBiS web applications. Next, we briefly describe the hardware-software configuration of our experiment and RUBiS.

*Hardware.* We use as the virtualized servers two HP Elite PCs, each with a Core i7-3770 CPU and 16 GB of memory. The CPU supports 15 speed levels using DVFS ranging from 1.6 to 3.41 GHz. In our experiment, we use six speed settings of $1.6, 1.9, 2.2, 2.6, 3$ and $3.4$ GHz and a zero speed to represent server shutdown. The client machine is a Dell Precision Desktop which has a Core i7 860@2.8GHz CPU and 4 GB of memory. The servers are powered through network enabled WattsUp power meter to measure the energy consumption remotely through a Java module placed in the client machine. The client machine is directly connected to the power outlet, hence the energy consumption of client machine is not captured by the power meter. In real system, eBud module will be placed in a control server belonging to the data center.

*Software.* XenServer 6.1 with free license is used in the servers as the hypervisor to host three VMs in each machine. Ubuntu Linux server 12.04 with SSH and LAMP server is installed in the VMs to facilitate hosting of the PHP version of RUBiS web service. Each server CPU has four processing cores[6], among which one is dedicated to XenServer and the remaining three cores, identified as VCPUs in XenServer, are distributed to the three VMs according to the CPU allocation decisions made by eBud. The VCPU allocations can take fraction values and the total allocation of the VMs in one server must remain equal or less than 3. The client machine runs with Windows 7 operating system and hosts eBud and RUBiS workload generator, both implemented using Java. XenServer java API is used to connect to the servers and issue VCPU assignment commands, while Apache SSH module is used to control the server speed.

*RUBiS.* RUBiS is a widely used performance benchmark application developed at Rice University [30], which imitates the behavior of an auction site (e.g., eBay). The implementation of RUBiS focuses on the core functionality of an auction site: selling, browsing and bidding. While there are several versions of RUBiS available, we use the PHP implementation in our experiment. The auction sites are hosted in the VMs and serve the user requests form the RUBiS workload generator implemented in the client machine. The RUBiS workload generator creates user sessions (a.k.a. clients) which simulate the user behaviors in an auction site. The number of clients indicates the workload being generated for the website.

*Other settings.* We use the same I/O traces of Microsoft Research [27] as in our simulation and scale it to have a maximum VM CPU utilization of 70 percent. To avoid lengthy experiment, the duration of a time slot in our experiment is 7 minutes and the total budgeting period is 48 time slots. Our energy budget is set to 2,995 W, which is 90 percent of the total energy consumption of capping-unaware algorithm and we use $\beta = 0.00006$. In the performance comparison part, the prediction window for PerfectPH algorithm is set to six time slots.

Although our experiment setup is scaled down from our theoretical formulation and simulation studies, it captures the long term energy budget problem in a virtualized environment, which is our main contribution in this work.

## 5.2 Experiment Results

In this subsection, we first present the server and VM performance models used in the experiment. We then compare eBud with the capping-unaware and PerfectPH algorithms, and finally show the impact of the cost-capping parameter $V$ in the experiment setup.

*Performance models.* In Fig. 9, we show the server and VM performance at different resource allocations and workload conditions. Fig. 9a shows the impact of server speed and VM CPU allocation on the VM service rate. We see that with increase of either of the two resources, speed and VM CPU, the service rate increases. The service rate in this figure is normalized with respect to the maximum service rate, which

6. Core i7-3770 processors support configuration of eight virtual cores with Intel hyper-threading technology. We disable hyper-threading in our experiment to avoid complexity in modeling the VM performance introduced by two layers (i.e software and hardware) of virtualization.

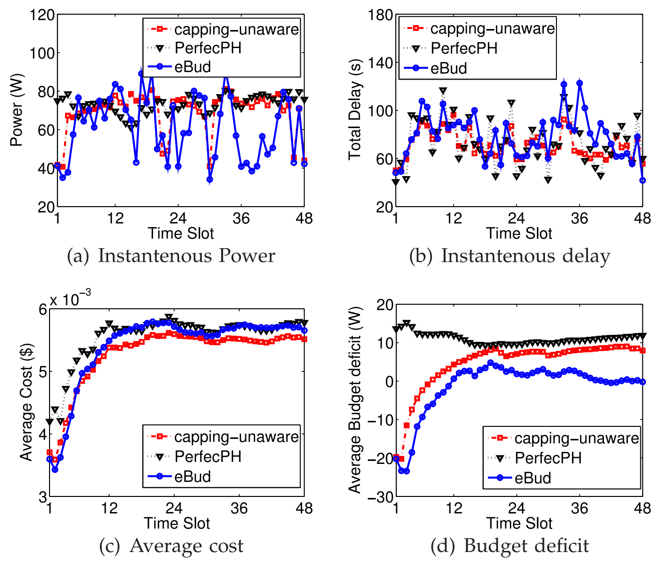


Fig. 10. Server and VM model for experiment.

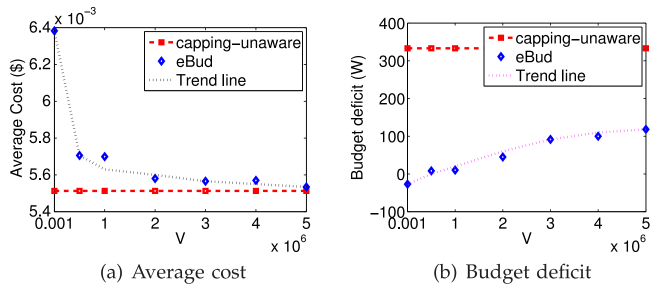


Fig. 11. Impact of V on average cost and budget.

is achieved at speed 3.4 GHz and 2.5 VCPU. In Fig. 9b, we see the server power consumption at different utilization levels for different speeds. While the power consumption increases almost linearly with utilization, we see a non-linear increase in power consumption with the increase in server speed (also observed in other studies [31]).

In Figs. 9c and 9d, we study the response time of the RUBiS website as we change the workload (i.e., number of clients) and CPU resources. In Fig. 9c, we see that with an increase in the number of clients, the average response time increases for different VCPU allocations. The sharp rise in the response time at these different VCPU allocations corresponds to the service rate (or service capacity) of the VM. Fig. 9d shows the change in the response time with increased server speed for different workload levels.

*Performance comparison.* In Fig. 10, we compare the power consumption and delay performance as well as the average cost and average budget deficit of eBud with the benchmark PerfectPH and the capping-unaware algorithms. Figs. 10a and 10b show the per-slot power and delay, respectively, for the three algorithms. We see that eBud has a lower power consumption compared to the other two algorithms without much impact on the delay performance. eBud consumes 10 and 16 percent less power compared to capping-unaware and PerfectPH algorithms. In Figs. 10c and 10d, we compare the average cost and average budget deficit of the three algorithms. We see that eBud only incurs 3.5 percent more cost than capping-unaware algorithm while manages to achieve a zero budget deficit while saving 10 percent energy. PerfectPH, on the other hand, incurs a higher cost than eBud while still missing the average per-slot budget by 11.85 W (even higher than the capping-unaware algorithm).

*Impact of V.* The control parameter V determines the trade-off between cost saving and meeting energy budgeting target and hence is a very important parameter for eBud. In Fig. 11, we show the impact of $V$ on the average cost and budget deficit for eBud in the experiment setup. We see that the experiment results match with the simulation studies presented in Fig. 3: with an increase in $V$, the average cost decreases while the budget deficit increases. eBud in these figure achieves a zero budget deficit at $V = 5 \times 10^5$ with a 3.5 percent more cost compared to the capping-unaware algorithm.

To sum up, we validate eBud and our analysis through the above experiments: Fig. 10 demonstrates the effective performance of eBud, and Fig. 11 captures the working principle of eBud and serves to validate the observations and findings of our simulation studies.

## 6 Related Work

We provide a snapshot of the related work from the following aspects.

*Data center optimization and VM resource management.* There has been a growing interest in optimizing data center operation from various perspectives such as cutting electricity bills [13], [14], [32] and minimizing response times [10], [19]. For example, "power proportionality" via dynamically turning on/off servers based on the workloads (a.k.a. dynamic capacity provisioning or right-sizing) has been extensively studied and advocated as a promising approach to reducing the energy cost of data centers [32]. As data centers are becoming increasingly virtualized, VM resource management has attracted much research interest: e.g., [33], [34] study optimum VM resource management in the cloud; [16] proposes admission control and dynamic CPU resource allocation to minimize the cost while bounding the queueing delay for batch jobs; [35] minimizes energy in a multi-tier virtualized environment with autonomic resource allocation; [36], [37], [38] study various dynamic VM placement and migration algorithms that may be combined with our proposed solution. These studies assume server CPU speed can be continuously chosen, which may not be practically realizable due to hardware constraints. Moreover, none of them have addressed the long term energy capping constraint.

*Power budgeting and energy capping.* Because it is very costly to increase the data center peak power (currently, estimated at 10-20 U.S. dollars per Watt) [10], optimally allocating the limited power budget to servers is crucial for performance improvement. In [10], the peak power budget is optimally allocated to (homogeneous) servers to minimize the total response time based on a queueing-theoretic model; [15] studies a similar problem but in the context of virtualized systems. Despite being a related study to power budgeting, "energy budgeting" or energy capping is relatively less explored. Recent studies, e.g., [2], [12], rely on long term prediction of the future information, which may not be feasible in practice. Similarly, [11] utilizes the prediction of long term future workloads to cap the monthly energy cost. While several heuristic algorithms (e.g., keep a schedule margin to offset the uncertainty in workload prediction) have been proposed in view of the unpredictable future information [2], their evaluation is empirical only, without providing any performance guarantees analytically. In comparison, eBud offers provable guarantees on the average cost while bounding the deviation from energy capping constraint, and our simulation results also demonstrate the benefits of eBud over the existing methods empirically. [39] studies energy budgeting for a data center, but it only considers dynamically turning on/off servers and hence does not apply to virtualized systems which require a set of holistic resource management decisions. Further, our study advances [39], [40] by proposing a richer set of resource management (including both workload distribution and server's power control) and also presenting a comprehensive evaluation of eBud via both simulations and scaled-down experiments.

## 7 Conclusion

In this paper, we studied energy budgeting for a virtualized data center and proposed an online algorithm, eBud, which determines the number of server to be turned on, VM resource allocation and workload distribution of different types of workloads for minimizing the data center operational cost while satisfying a long term energy capping constraint. It was proved that eBud achieves a close-to-minimum operational cost compared to the optimal offline algorithm with future information, while bounding the potential violation of energy budget constraint. We performed a trace-based simulation study to complement the analysis. The results show that eBud reduces the cost by more than 16 percent (compared to state-of-the-art prediction-based method) while resulting in the same energy consumption. Finally, we support our simulation study through experiment which shows the effectiveness of eBud in real life.

## Appendix

We now provide an outline for the proof that follows the recently-developed sample-path Lyapunov optimization technique [25]. We first introduce a family of offline algorithm that we will compare eBud with. Specifically, we divide the entire budgeting period into $R$ frames, each having $T \geq 1$ time slots, such that $K = RT$. There exists an oracle that has the complete information over the entire frame (i.e., $T$ time slots) at the beginning of each frame. Then, at the beginning of the $r$-th frame, for $r = 0, 1, \ldots, R-1$, the oracle chooses a sequence of decisions to solve the following problem:

$$\mathbf{P3}: \quad \min_{\mathbf{m}(t),\mathbf{x}(t),\lambda(t)} \frac{1}{T} \sum_{t=rT}^{(r+1)T-1} g(\mathbf{m}(t), \mathbf{x}(t), \lambda(t)) \tag{20}$$

$$s.t., \quad \text{constraints } (10), (11), (12), (13), (14), \tag{21}$$

$$\sum_{t=rT}^{(r+1)T-1} p(\mathbf{u}(t), \mathbf{m}(t)) \leq \frac{Z}{R}. \tag{22}$$

We assume that **P3** admits at least one feasible solution and denote the minimum average cost for the $r$-th frame by $G_r^*$, for $r = 0, 1, \ldots, R-1$, and hence, the long term minimum average cost achieved by the oracle's optimal $T$-step lookahead algorithm is given by $\frac{1}{R}\sum_{r=0}^{R-1} G_r^*$. Then, Theorem 1 can be proved following three key steps (whose details can be found in [25]):

1) We relate the energy deficit queue length to approximate constraint satisfaction.
2) We define a quadratic Lyapunov function for the energy deficit queue length and derive upper bounds on the one-slot as well as $T$-slot Lyapunov drift plus cost.
3) We minimize the derived upper bounds using eBud and then compare with the optimal offline algorithm with $T$-step lookahead information to complete the proof.

Through the above steps, it will be seen that the online optimization problem in **P2** is actually equivalent to minimizing the derived upper bound on the $T$-slot Lyapunov drift plus cost. Then, by substituting the optimal offline solution to **P3** into the expression of Lyapunov drift plus cost and through simple mathematical manipulations, we can obtain a bound on the difference between eBud and the optimal offline solution to **P3**.

## Acknowledgments

This work was supported in part by NSF CNS-1018108 and CNS-1423137.

## References

[1] Virtualizing the Enterprise: An overview. [Online]. Available: http://www.zdnet.com/virtualizing-the-enterprise-an-overview-7000018110 /, 2013.

[2] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, and M. Martonosi, "Capping the brown energy consumption of internet services at low cost," in *Proc. Int. Green Comput. Conf.*, 2010, pp. 3–14.

[3] Uptime Institute. (2013). Data center industry survey. [Online]. Available: http://uptimeinstitute.com/2013-survey-results

[4] U.S. Green Building Council, "Leadership in energy & environmental design," [Online]. Available: http://www.usgbc.org/leed

[5] California energy efficiency goals and potential studies. [Online]. Available: http://www.cpuc.ca.gov/PUC/energy/Energy+Efficiency/Energy+Efficiency+Goals+and+Potential+Studies.htm

[6] U.S. Federal Leadership in Environmental, Energy and Economic Performance—EXECUTIVE ORDER 13514. [Online]. Available: http://www.whitehouse.gov/administration/eop/ceq/sustainability

[7] Google, "Google's green PPAs: What, how, and why," http://static.googleusercontent.com/external_content/untrusted_dlcp/cfz.cc/en/us/green/pdfs/renewable-energy.pdf

[8] T. DiCaprio, "Becoming carbon neutral: How microsoft is striving to become leaner, greener, and more accountable," *Microsoft Whitepaper*, Jun. 2012.

[9] Uptime Institute, "2014 data center industry survey results," [Online]. Available: http://symposium.uptimeinstitute.com/images/stories/symposium2014/prese ntations/mattstansberry-surveyresults2014.pdf.

[10] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *Proc. 11th Int. Joint Conf. Meas. Model. Comput. Syst.*, 2009, pp. 157–168.

[11] Y. Zhang, Y. Wang, and X. Wang, "Electricity bill capping for cloud-scale data centers that impact the power markets," in *Proc. 41st Int. Conf. Parallel Process.*, 2012, pp. 440–449.

[12] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, "Carbon-aware energy capacity planning for datacenters," in *Proc. IEEE 20th Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2012, pp. 391–400.
[13] L. Rao, X. Liu, L. Xie, and W. Liu, "Reducing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.
[14] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst.*, 2011, pp. 233–244.
[15] H. Lim, A. Kansal, and J. Liu, "Power budgeting for virtualized data centers," in *Proc. USENIX Annu. Tech. Conf.*, 2011, p. 5.
[16] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp.*, 2010, pp. 479–486.
[17] S. Ghiasi, T. Keller, and F. Rawson, "Scheduling for heterogeneous processors in server systems," in *Proc. 2nd Conf. Comput. Front.*, 2005, pp. 199–210.
[18] S. Liu, S. Ren, G. Quan, M. Zhao, and S.-P. Ren, "Profit-aware load balancing for distributed cloud data centers," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process.*, 2013, pp. 611–622.
[19] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *Proc. Int. Green Comput. Conf.*, 2012, pp. 1–10.
[20] S. Kundu, R. Rangaswami, A. Gulati, K. Dutta, and M. Zhao, "Modeling virtualized applications using machine learning techniques," in *Proc. 8th ACM SIGPLAN/SIGOPS Conf. Virtual Execution Environ.*, 2012, pp. 3–14.
[21] L. Wang, J. Xu, and M. Zhao, "Modeling VM performance interference with fuzzy MIMO model," in *Proc. Int. Workshop Feedback Comput.*, 2012.
[22] N. U. Prabhu, *Foundations of Queueing Theory*. Norwell, MA, USA: Kluwer, 1997.
[23] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proc. USENIX Annu. Tech. Conf.*, 2009, p. 28.
[24] A. Gandhi, M. Harchol-Balter, R. Raghunathan, and M. A. Kozuch, "Autoscale: Dynamic, robust capacity management for multi-tier data centers," *ACM Trans. Comput. Syst.*, vol. 30, no. 4, pp. 14:1–14:26, Nov. 2012.
[25] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool, 2010.
[26] S. Boyd, A. Ghosh, and A. Magnani. (2003). Branch and bound methods. [Online]. Available: http://www.stanford.edu/class/ee392o/bb.pdf
[27] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. IEEE INFOCOM*, 2011, pp. 1098–1106.
[28] Q. Wu, "Making facebook's software infrastructure more energy efficient with autoscale," 2014.
[29] California ISO. [Online]. Available: http://www.caiso.com/
[30] Rubis: Rice University Bidding System. [Online]. Available: http://rubis.ow2.org/
[31] J. R. Lorch and A. J. Smit, "Improving dynamic voltage scaling algorithms with pace," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 2001, pp. 50–61.
[32] B. Guenter, N. Jain, and C. Williams, "Managing cost, performance and reliability tradeoffs for energy-aware server provisioning," in *Proc. IEEE INFOCOM*, 2011, pp. 1332–1340.
[33] Q. Zhu and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments," *IEEE Trans. Serv. Comput.*, vol. 5, no. 4, pp. 497–511, Fourth Quarter 2012.
[34] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 5, no. 2, pp. 164–177, Apr. 2012.
[35] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang, "Energy-aware autonomic resource allocation in multitier virtualized environments," *IEEE Trans. Serv. Comput.*, vol. 5, no. 1, pp. 2–19, Jan. 2012.
[36] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Proc. IEEE INFOCOM*, 2011, pp. 66–70.
[37] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. IEEE/ACM Int'l Conf. Green Comput. Commun./Int. Conf. Cyber, Phys. Social Comput.*, 2010, pp. 179–188.
[38] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *Proc. 20th Int. Symp. High Perform. Distrib. Comput.*, 2011, pp. 171–182.
[39] H. Mahmud and S. Ren, "Online resource management for data center with energy capping," presented at the 8th Int. Workshop Feedback Comput., San Jose, CA, USA, 2013.
[40] M. A. Islam, S. Ren, and G. Quan, "Online energy budgeting for virtualized data centers," in *Proc. IEEE 21st Int. Symp. Model., Anal. Simul. Comput. Telecommun. Syst.*, 2013, pp. 424–433.

**Mohammad A. Islam** received the BSc degree in electrical and electronics engineering from the Bangladesh University of Engineering and Technology in January 2008. He is currently working toward the PhD degree in electrical engineering at Florida International University (FIU), Miami. He is a member of the Sustainable Computing Group (SCG) at FIU led by Dr. S. Ren. His research interests include data center resource management, cloud computing, and sustainability for IT.

**Shaolei Ren** received the BE, MPhil, and PhD degrees, all in electrical engineering, from Tsinghua University in July 2006, Hong Kong University of Science and Technology in August 2008, and University of California, Los Angeles, in June 2012, respectively. Since August 2012, he has been with the School of Computing and Information Sciences, Florida International University, Miami, as an assistant professor. His research interests include cloud computing, data center resource management, and sustainable computing. He received the Best Paper Award from IEEE International Conference on Communications in 2009 and from International Workshop on Feedback Computing in 2013.

**A. Hasan Mahmud** received the BSc degree in computer science and engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in January 2008. He is working toward the PhD degree at the School of Computing and Information Sciences, Florida International University, Miami. Before joining Florida International University, he spent 3.5 years in a leading software company in Bangladesh. He is currently working in the Sustainable Computing Group, Florida International University, led by Dr. S. Ren. His current research interests include resource management in cloud computing, capacity provisioning, and autoscaling of virtualized resources. He has received the Best Paper Award in 8th International Workshop in Feedback Computing, 2013.

**Gang Quan** (M'02-SM'10) received the BS degree from Tsinghua University, Beijing, China, the MS degree from the Chinese Academy of Sciences, Beijing, and the PhD degree from the University of Notre Dame, Notre Dame, Indiana. He is currently an associate professor with the Electrical and Computer Engineering Department, Florida International University, Miami. His research interests include real-time system, power/thermal aware design, embedded system design, advanced computer architecture, and reconfigurable computing. He is the recipient of a National Science Foundation Faculty Career Award. He also received the Best Paper Award from the 38th Design Automation Conference. His paper was also selected as one of the Most Influential Papers of 10 Years Design, Automation, and Test in Europe Conference (DATE) in 2007. He is a senior member of IEEE.