Online Energy Budgeting for Virtualized Data Centers

Mohammad A. Islam
Florida International University
Email: misla012@fiu.edu

Shaolei Ren
Florida International University
Email: sren@cis.fiu.edu

Gang Quan
Florida International University
Email: gang.quan@fiu.edu

***Abstract*—Increasingly serious concerns about the IT carbon footprints have been pushing data center operators to cap their (brown) energy consumption. Naturally, achieving energy capping involves deciding the energy usage over a long timescale (without foreseeing the far future) and hence, we call this process "*energy budgeting*". The specific goal of this paper is to study energy budgeting for virtualized data centers from an *algorithmic* perspective: we develop a provably-efficient online algorithm, called eBud (energy Budgeting), which determines server CPU speed and resource allocation to virtual machines for minimizing the data center operational cost while satisfying the long-term energy capping constraint in an online fashion. We rigorously 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 analysis. The simulation results are consistent with our theoretical analysis and show that eBud reduces the cost by more than 60% (compared to state-of-the-art prediction-based algorithm) while resulting in a zero energy budget deficit.**

I. Introduction

Driven by the demand for scalable and robust provision of computing services, virtualization has found its successful usage in a wide spectrum of data center systems, e.g., Amazon EC2 and Microsoft Azure. Despite the advantage of enabling "power proportionality" using virtualization, the continuous growth of virtualized data centers is still accompanied by surging electricity consumption (often labeled as "brown energy" due to its carbon-intensive sources), raising serious concerns about their carbon footprints and environmental impacts. Consequently, data center operators are nowadays constantly pressured to cap the brown energy consumption, either mandated by governments in the form of Kyoto-style protocols or required by utility companies [1]–[5].

While a significant progress has been achieved in reducing the carbon footprint (e.g., [1], [5], [6]), it still remains a challenging task to cap the actual long-term (e.g., monthly or yearly) brown energy consumption of data centers, which has become increasingly important for sustainable computing in the future. In addition to the appealing sustainability, capping the long-term brown energy also delivers other benefits such as tax reduction, favorable accreditation, and/or better bargaining power with utility companies [1], [2], [5]. Furthermore, if offset by an equivalent amount of renewable energy credits or RECs, capping the brown energy consumption equates the trending *carbon neutrality* or "*net-zero*", which has become the long-term strategic goal that many large IT companies such as Google and Microsoft pledge to achieve [1], [2]. Naturally, achieving energy capping involves deciding the energy usage over a long term and hence, we call this process "*energy budgeting*", in comparison with power budgeting that allocates the peak power to different servers in a data center and has been well studied [7], [8]. Nonetheless, energy budgeting is still relatively less explored as it faces a significant challenge that power budgeting does not have: data center operator needs to decide its energy usage in an online manner that cannot possibly foresee the far future time-varying workloads. While some initial efforts have been made to achieve energy capping for data centers [3]–[5], they require accurate prediction of long-term future workloads that is typically unavailable in practice (e.g., due to unpredicted traffic spikes). Moreover, to our best knowledge, energy budgeting for virtualized data centers has not been studied by prior work.

In this paper, we study energy budgeting for virtualized data centers and propose a provably-efficient online algorithm, called eBud (energy Budgeting), which determines server CPU speed and virtual machine (VM) resource allocation for minimizing the data center operational cost while satisfying the long-term energy capping constraint. eBud can be integrated into the existing resource management module in virtualized data centers and implemented in an online manner based on *short*-term workload prediction. Building upon the recently developed Lyapunov optimization technique [9], we rigorously prove that compared to the optimal offline algorithm with lookahead information, eBud achieves a close-to-minimum cost while *approximately* achieving the energy capping constraint with a bounded "fudge" factor. 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 60% while satisfying the long-term energy budget constraint. Sensitivity studies also demonstrate that eBud is robust against various factors such as inaccurate knowledge of VM service rate.

The rest of this paper is organized as follows. Section II describes the model. In Section III and IV, we present

1526-7539/13 $26.00 © 2013 IEEE
DOI 10.1109/MASCOTS.2013.64

the problem formulation and develop our online algorithm, eBud. Section V provides a simulation study. Related work is reviewed in Section VI and finally, concluding remarks are offered in Section VII.

II. Model

We consider a discrete time model by equally dividing the total budgeting period (e.g., a month) into K time slots indexed by $t = 0, 1, \cdots, K-1$. The system block diagram, illustrating the decisions and integration of eBud in the existing system, is shown in Fig. 1. Next, we present the modeling details for the data center, VMs and workloads.

A. Data Center

We consider a data center with M physical servers hosting VMs. Without causing ambiguity, we also use *servers* to represent physical servers wherever applicable. The servers are possibly heterogeneous in their power consumption and performance, and each server may trade its performance for power consumption by varying its processing speed (e.g., via dynamic voltage and frequency scaling or DVFS). As computing takes up a significant portion (typically 40%) of server power consumption [7], we focus on CPU resource allocation, while treating other resources (e.g., memory, disk) as sufficient and *non-bottleneck* resources that consume a relatively *constant* power. 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 [10], [11].

To keep our model general, we consider that server i can choose its speed $x_i(t)$ out of a finite set $S_i = \{s_{i,0}, s_{i,1}, \cdots, s_{i,L_i}\}$ where $s_{i,0} = 0$ represents zero speed (server deep sleep or shut down) and L_i is the number of available positive speed settings. The speed $x_i(t)$ quantifies server i's total CPU resource (measured in, e.g., GHz) that is available for processing VM workloads (besides the CPU consumption by the privileged domain0/root VM handling resource management). Assuming that the servers consume a negligible power under the zero-speed mode, we express the average power consumption and peak power of server i as [10]

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

$$\hat{p}_i(x_i) = p_{i,s} + p_{i,c}(x_i), \quad (2)$$

where u_i is CPU utilization of server i (that will be specified in the next section), $p_{i,s}$ is the static power regardless of the workloads as long as server i is turned on, $p_{i,c}(x_i)$ is the computing power incurred only when server i is operating at a speed of x_i, and the indicator function $\mathbf{1}_{(x_i>0)} = 1$ if and only if the processing speed $x_i > 0$. In our study, we only focus on the server power consumption for the considered workloads, while neglecting the power consumption of other parts (e.g., power supply system, cooling system) which however can be conveniently absorbed by a (time-varying) power usage effectiveness (PUE) factor that, multiplied by the server power consumption, yields the total data center power consumption [6]. Thus, the total power consumption[1] during time t is given by

$$p(\mathbf{u}(t), \mathbf{x}(t)) = \sum_{i=1}^{M} p_i(u_i(t), x_i(t)), \quad (3)$$

where $\mathbf{u}(t) = (u_1(t), u_2(t), \cdots, u_M(t))$ and $\mathbf{x}(t) = (x_1(t), x_2(t), \cdots, x_M(t))$ are the vectors of utilization and speed selections, respectively. Note that, as in [6], [12], we ignore the possible *toggling* costs incurred when updating decisions (e.g., turning a server off or into deep sleep, VM migration) that can be dealt with using techniques developed in [13].

We denote the electricity price at time t by $w(t)$, which is known to the data center no later than the beginning of time t and may change over time if the data center participates in a real-time electricity market (e.g., hourly market [6], [12]). At time t, the incurred electricity cost can be expressed as

$$e(\mathbf{u}(t), \mathbf{x}(t)) = w(t) \cdot p(\mathbf{u}(t), \mathbf{x}(t)). \quad (4)$$

While we use (4) to represent the electricity cost for the data center at time t (as considered by [6], [12]),our analysis is not restricted to a linear electricity cost function and can also model other electricity cost functions such as nonlinear convex functions (e.g., the data center is charged at a higher price if it consumes more power).

B. Virtual Machine

Virtualization provides abstraction of the underlying hardware to upper layers by creating a set of virtualized system platforms on which a customized operating system can run. In a virtualized environment, each physical 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. There are totally N types of workloads, each of which is processed by a dedicated VM. We refer to the VM processing type-j jobs as VM j, for $j = 1, 2, \cdots, N$. Server i hosts a subset of VMs denoted by $\mathcal{N}_i \in \mathcal{N}$, such that $\mathcal{N}_1 \times \mathcal{N}_2 \times \cdots \times \mathcal{N}_M = \{1, 2, \cdots, N\}$ and $\mathcal{N}_i \cap \mathcal{N}_j = \varnothing$ if $i \neq j$, i.e., all VMs are hosted on physical servers and no VM is simultaneously hosted on two physical servers. We also use a M-by-N matrix $\mathbf{A}$ of binary values $a_{i,j}$ to indicate the workload/VM placement decision that maps workloads/VMs to physical servers: $a_{i,j} = 1$ if and only if server i hosts VM j, i.e., $j \in \mathcal{N}_i$, for $i = 1, 2, \cdots, M$ and $j \in \mathcal{N}$. VM j hosted on server i is allocated an amount of CPU resources quantified by $c_{i,j}$, while the other type of hardware resources (e.g., memory,

[1]This is equivalent to energy consumption, since the length of each time slot is the same.

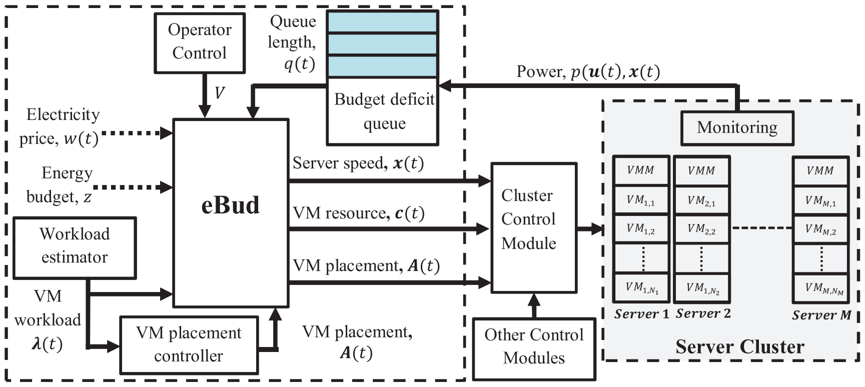


Figure 1. System block diagram.

disk) are non-bottleneck for our considered CPU-intensive workloads [10]. For notational convenience, we also use the matrix expression $\mathbf{c} = [c_{i,j}]_{i=1,\cdots,M,j=1,\cdots N}$.

C. Workload

We focus on delay-sensitive CPU-intensive workloads/jobs (as in [6], [10]), whereas delay-tolerant batch workloads can be easily captured by maintaining a separate batch job queue as considered by several existing studies [10]. There are N types of workloads, and we denote by $\lambda_j(t) \in [0, \lambda_{j\max}]$ the arrival rate of type-j workloads during time t. "Zero" arrival rate means there is no certain type of workloads. As assumed in prior work [3], [6], [13], the value of $\lambda_j(t)$ is accurately available at the beginning of each time slot t, as can be achieved by using various techniques such as regression analysis.

The service rate for type-j workload (i.e., how many jobs can be processed in a unit time) is given by $\mu_j = \mu_{i,j}(c_{i,j})$, where $c_{i,j}$ is the amount of CPU resources that server i allocates to VM j, and 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 $\mu_{i,j}(\cdot)$ [14], [15], 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) [11]. Thus, as in [10], [16], we assume in the following analysis that $\mu_{i,j}(\cdot)$ is exogenously determined, for $i = 1, 2, \cdots, M$ and $j = 1, 2, \cdots, N$, while noting that modeling the function $\mu_{i,j}(\cdot)$ can be done using the techniques developed in [14], [15] that are beyond the scope of our study.

To quantize the data center performance, we introduce the *delay cost*: we denote the delay cost for type-j workloads by a function $d_j(\lambda_j, \mu_j)$, which is intuitively increasing in λ_j and decreasing in μ_j [6], [13]. As a concrete example, 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 $\frac{1}{\mu_j - \lambda_j}$ [17] and hence, the total delay cost at time t can be written as

$$d(\lambda(t), \mu(t)) = \sum_{j=1}^{N} \frac{\lambda_j(t)}{\mu_j(t) - \lambda_j(t)}, \tag{5}$$

in which $\lambda(t) = (\lambda_1(t), \cdots, \lambda_N(t))$, $\mu(t) = (\mu_1(t), \cdots, \mu_N(t))$, and we ignore the network delay cost that can be approximately modeled as a certain constant [6] and added into (5) without affecting our approach of analysis. Note that other delay cost functions, which exploit workload characteristics and data center architectures to better estimate the delay performance, can also be used without affecting our approach of solution. Finally, we note that the delay cost model in (5) implicitly assumes that VMs are *perfectly* isolated without interfering with each other. While this may not be true in heavy traffic regimes (e.g., due to cache, I/O contention) as pointed out by the existing research [14], we consider perfect isolation and use the delay cost model only as an *approximate* indication for the actual delay performance (as studied in [10], [16]). In other words, the model is only intended to facilitate online resource management, 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.

III. Problem formulation

In this section, we specify the optimization objective and constraints. Then, we present an offline formulation for the energy budgeting problem as well as a T-step lookahead algorithm that we compare eBud with.

A. Objective and Constraint

At the beginning of each time slot, the data center updates CPU speed selection for each server and VM resource allocation to minimize the operational cost subject to a set of constraints as specified below.

Objective: We focus on minimizing operational cost of the data center rather than capital cost (e.g. building data centers). In general, both electricity cost and delay cost are important for data centers, as the former takes up a dominant fraction of the operational cost while the later affects the user experiences and revenues [13]. We incorporate both costs by constructing a parameterized cost function as follows

$$g(\mathbf{x}(t), \mathbf{c}(t)) = e(\mathbf{u}(t), \mathbf{x}(t)) + \beta \cdot d(\lambda(t), \mu(t)), \qquad (6)$$

where both the server utilization vector $\mathbf{u}(t)$ and service rate vector $\mu(t)$ are functions of the VM resource allocation $\mathbf{c}(t)$, and $\beta \geq 0$ is the weighting parameter for delay cost relative to the electricity energy cost [6], [13]. The optimization objective is to minimize the long-term average cost expressed as

$$\bar{g} = \frac{1}{K} \sum_{t=0}^{K-1} g(\mathbf{x}(t), \mathbf{c}(t)), \qquad (7)$$

where K is the total number of time slots over the entire budgeting period (e.g., a month).

Constraints: Naturally, server i can only select one of the supported speeds, i.e.

$$x_i(t) \in \mathcal{S}_i = \{s_{i,0}, s_{i,1}, \cdots, s_{i,L_i}\}, \ \forall i, t. \qquad (8)$$

The power distribution system (consisting of the power supply/distribution units, failure backup generator, etc.) of a data center is typically subject to a *predetermined* maximum operating power capacity that cannot be easily increased without hardware expansion. In addition, data centers are often partly billed based on their peak power usage, although the peak usage charge is not explicitly taken into consideration in our study. Thus, in light of that exceeding the peak power limit results in fines and/or increased financial charges, we capture in our formulation the peak power constraint as follows

$$\sum_{i=1}^{M} \hat{p}_i(x_i(t)) \leq \hat{P}, \qquad (9)$$

where $\hat{P}$ is the peak (server) power constraint.

To avoid server overloading and workload dropping, at any time t, the VM resource allocation needs to satisfy

$$\sum_{j \in \mathcal{N}_i} c_{i,j}(t) \leq x_i(t), \ \forall i, \qquad (10)$$

$$\lambda_j(t) \leq \theta \cdot \mu_j(t), \ \forall j, \qquad (11)$$

where $\theta \in (0,1)$ specifies the maximum utilization of a single VM and $\mu_j(t) = \mu_{i,j}(c_{i,j}(t))$ maps the CPU allocation to the service rate for type-j workloads processed by server i. Note that additional constraints, such as discreteness constraints for CPU resource allocations (i.e., $c_{i,j}$ can only be chosen out of a discrete set), can also be easily incorporated in our study. Now, we are ready to derive the average server utilization $u_i(t)$ as follows

$$u_i(t) = \sum_{j \in \mathcal{N}_i} \frac{\lambda_j(t)}{\mu_j(t)} \cdot \frac{c_{i,j}(t)}{x_i(t)}, \qquad (12)$$

where $\frac{\lambda_j(t)}{\mu_j(t)}$ is the utilization of VM j and $\frac{c_{i,j}(t)}{x_i(t)}$ is portion of server i's CPU resources allocated to VM j. This completes the expression of average server power consumption in (1).

Next, assuming that electricity energy is "brown", we specify the brown energy capping constraint as follows

$$\frac{1}{K} \sum_{t=0}^{K-1} p(\mathbf{u}(t), \mathbf{x}(t)) \leq \frac{Z}{K}, \qquad (13)$$

where Z is the (desired) capping constraint over the entire budgeting period. Note that our study can also address the scenario in which part of the electricity is produced by green energy sources: by multiplying the electricity usage with a certain factor that indicates the percentage of "brown" electricity, (13) specifies the desired upper limit on the actual "brown" electricity usage.

Before proceeding with the algorithm design, we emphasize that our main focus is on deciding CPU speed for each server and allocating CPU resources to VMs, while treating the workload/VM placement decision $\mathbf{A}(t)$ as exogenously given. Many proposals that address VM placement from various perspectives (e.g., application-aware VM placement [18], multi-objective-based VM placement [19]) as well as dynamic VM migration [20] can be used to make the VM placement decision $\mathbf{A}(t)$ and incorporated into our study. While we isolate the VM placement decision as in the related literature (e.g., power budgeting [7]), combining the VM placement with our proposed eBud is naturally expected to further reduce the operational cost of data centers, thereby pointing to a potential research direction that may be pursued in our future work.

B. *Offline problem formulation*

This subsection presents an offline problem formulation for the resource management as follows:

$$\textbf{P1}: \quad \min_{\mathcal{H}} \bar{g} = \frac{1}{K} \sum_{t=0}^{K-1} g(\mathbf{x}(t), \mathbf{c}(t)) \qquad (14)$$

$$s.t., \quad \text{constraints } (8), (9), (10), (11), (13), \qquad (15)$$

where $\mathcal{H}$ represents a sequence of decisions, i.e., $\mathbf{x}(t), \mathbf{c}(t)$, for $t = 0, 1, \cdots, K-1$, which we need to optimize. Optimally solving **P1** requires complete offline information (i.e., workload arrivals and electricity prices) over the entire budgeting period that is very difficult, if not impossible, to accurately predict in advance, especially in view of the frequent traffic surges due to breaking events that can significantly boost the workloads [21], [22], as further corroborated by our university data center trace. Moreover, due to the discreteness of server speeds, **P1** involves a *long* sequence of mixed-integer nonlinear programming problems and is difficult to solve, even if the long-term future information is accurately known a priori. Next, we propose an efficient online algorithm (Section IV) to address the challenges.

T-step lookahead algorithm. As a benchmark, we introduce a family of offline algorithm parameterized by the lookahead information window size T. 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 $r = 0, 1, \cdots, R-1$, the oracle chooses a sequence of decisions to solve the following problem:

$$\textbf{P2}: \quad \min_{\mathbf{x}(t),\mathbf{c}(t)} \frac{1}{T} \sum_{t=rT}^{(r+1)T-1} g(\mathbf{x}(t), \mathbf{c}(t)) \tag{16}$$

$$s.t., \quad \text{constraints (8), (9), (10), (11),} \tag{17}$$

$$\sum_{t=rT}^{(r+1)T-1} p(\mathbf{u}(t), \mathbf{x}(t)) \leq \frac{Z}{R}, \tag{18}$$

To ensure there exists at least one feasible solution to **P2**, we make the two assumptions that are very mild in practice.

Boundedness assumption: The workload arrival rate $\lambda(t)$, is finite, for $t = 0, 1, \cdots, K-1$.

Feasibility assumption: For the r-th frame, where $r = 0, 1, \cdots, R-1$, there exists at least one sequence of resource management decisions that satisfy the constraints of **P2**.

The boundedness assumption, combined with (8), ensures that the cost function is finite, while the feasibility assumption guarantees that the oracle can make a sequence of feasible decisions to solve **P2**. We denote the minimum average cost for the r-th frame by G_r^*, for $r = 0, 1, \cdots, R-1$, and hence, the long-term minimum average cost achieved by the oracle's optimal T-step lookahead algorithm is given by $\frac{1}{R}\sum_{r=0}^{R-1} G_r^*$.

IV. Online Energy Budgeting

In this section, we present an online algorithm, eBud, which is provably efficient in terms of cost minimization compared to the optimal offline algorithm with T-step lookahead information. Building upon yet extending the recently developed Lyapunov optimization technique [9], eBud only uses online information and allows the data center operator to adaptively adjust the tradeoff between cost saving and how much the electricity usage violates the long-term energy capping constraint.

A. eBud

The long-term energy budget constraint couples together the resource management decisions across all the time slots spanning the budgeting period, thereby requiring complete future information (i.e. workload arrival rates and electricity price during the whole budgeting period) to make the optimal decisions. Nonetheless, as workload and electricity price are online information and may not be available *a priori*, the decision coupling across time slots renders offline solutions practically infeasible, even though the intolerable complexities are ignored. To tackle this challenge, in eBud, we transform the long-term energy budget constraint (13) to a perturbing term in the objective function to enable online decisions. Specifically, building upon the Lyapunov optimization technique [9], we construct a (virtual) energy budget deficit queue which acts as a guiding mechanism to meet the long-term budget constraint: assuming $q(0) = 0$, energy budget the deficit queue dynamics evolves as follows

$$q(t+1) = \left[q(t) + p(\mathbf{u}(t), \mathbf{x}(t)) - \frac{Z}{K}\right]^+, \tag{20}$$

where $q(t)$ is the queue length indicating how far the current electricity usage deviates from the average budget. 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.

Algorithm 1 eBud

```
1: Input $\lambda(t)$ and $w(t)$ at the beginning of each time $t = 0, 1, \cdots, K-1$
2: if $t = rT$, $\forall r = 0, 1, \cdots, R-1$ then
3:     $q(t) \leftarrow 0$ and $V \leftarrow V_r$
4: end if
5: Choose $\mathbf{x}(t)$ and $\mathbf{c}(t)$ subject to (8),(8),(10),(11) to minimize
       P3 :  $V \cdot g(\mathbf{x}(t), \mathbf{c}(t)) + q(t) \cdot p(\mathbf{u}(t), \mathbf{x}(t))$     (19)
6: At the end of time slot, update $q(t)$ according to (20).
```

eBud is a an online algorithms which only requires the currently available information (i.e. $\lambda(t)$, $w(t)$) as inputs. We use $V_1, V_2, \cdots, V_R$ 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 CPU resource allocation 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.

Working principle of eBud. The main idea of eBud is to keep track of the deviation from the long-term target and tweak the objective function accordingly to gradually nullify the deviation. Specifically, the objective function in eBud contains the original cost function (6) scaled by V plus the perturbing term (i.e., energy consumption scaled by the carbon deficit queue $q(t)$). The cost-capping parameter V determines the relative weight of cost minimization. In particular, a smaller V indicates that the budget deficit queue has a greater effect on **P3** and excessive energy consumption is recovered in fewer time slots. On the other hand, a greater V will push the algorithm closer towards cost minimization with less concerns with the long-term energy capping constraint. In general, the appropriate value of V depends on the specific modeling parameters and are typically determined by experience and on a trial-and-error basis. For example, if the current cost is too high whereas the electricity usage is far below the energy budget, the data center operator may increase the value of V to weaken the impact of energy deficit queue. The impact of V on data center operation will be further formalized in our algorithm analysis as well as in the simulation.

eBud in high workloads. In eBud, we do not put any *hard* constraint on the long-term energy consumption. We instead incorporate a feedback mechanism which guides the optimization towards meeting the long-term budget. One important aspect of using this technique is that it prevents workload dropping and significantly degraded delay performance in the case of high workloads: if the energy budget is insufficient, a higher priority is given to processing workloads rather than satisfying the energy capping constraint. Another way of providing a guaranteed delay performance is that we impose an additional delay constraint in the optimization process, and doing so will only introduce an additional constraint in **P3** without affecting our online mechanism.

Finally, it is worth mentioning that **P3** is still a mixed-integer problem. However, although the complexity of **P3** is exponential in the number of servers, eBud is practically realizable, because the resource management decision is only made once every time slot (i.e., the total *intolerable* complexity is amortized over each time slot).

B. Performance analysis

By extending the standard Lyapunov optimization technique [9], this subsection presents the performance analysis of eBud in Theorem 1, whose proof is available in [23].

Theorem 1. *Suppose that boundedness and feasibility assumptions are satisfied. Then, 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:*

$$\frac{1}{K}\sum_{t=0}^{K-1} p\left(\mathbf{u}(t), \mathbf{x}(t)\right) \leq \frac{Z}{K} + \frac{\sum_{r=0}^{R-1}\sqrt{C(T) + V_r\left(G_r^* - g_{\min}\right)}}{R\sqrt{T}}, \tag{21}$$

where $C(T) = B + D(T-1)$ with B and D being finite constants defined in [23], G_r^* is the minimum average cost achieved over the r-th frame by the optimal offline algorithm with T-slot lookahead information, for $r = 0, 1, \cdots, R-1$, and $g_{\min}$ is the minimum hourly cost that can be achieved by any feasible decisions throughout the budgeting period.

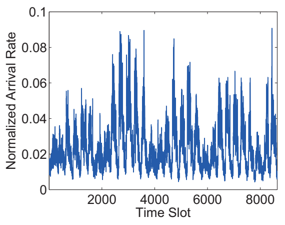
b. *The average cost $\bar{g}^*$ achieved by eBud satisfies:*

$$\bar{g}^* \leq \frac{1}{R}\sum_{r=0}^{R-1} G_r^* + \frac{C(T)}{R} \cdot \sum_{r=0}^{R-1}\frac{1}{V_r}. \tag{22}$$

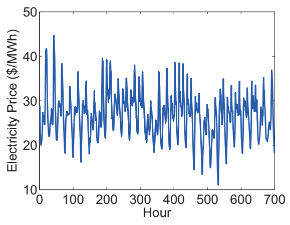
Theorem 1 shows that, given a fixed value of T and R, eBud is $O(1/V)$-optimal with respect to the average cost against the optimal T-step lookahead policy, i.e., eBud incurs no more than $O(1/V)$ additive cost than the minimum value, while the energy capping constraint is guaranteed to be <u>approximately</u> satisfied with a bounded "fudge factor" of $\frac{\sum_{r=0}^{R-1}\sqrt{C(T)+V_r(G_r^*-g_{\min})}}{R\sqrt{T}}$. With a larger V, the cost is closer to the minimum but the potential deviation of electricity usage from the capping constraint can be larger, and vice versa. We will consider case studies using realistic settings to substantiate this statement by providing more accurate estimates of costs and electricity usage. Note that additional assumptions on the environment dynamics (e.g., i.i.d./Markovian workload arrival rate and electricity price [9]) can be made to derive tighter analytical bounds.

C. 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 server processing speed as well as VM CPU resource allocation and then forwards the VM placement decisions, supplementary control modules (i.e. cooling system control, non-CPU VM resource allocation) can be appended after eBud to improve the performance. Any existing control module that made overlapping decisions are placed before eBud, and can be adopted as additional constraints in our algorithm.

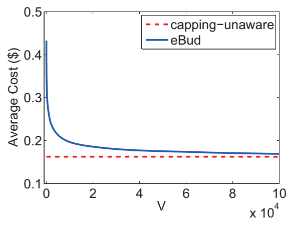


(a) Workload of VM-1.

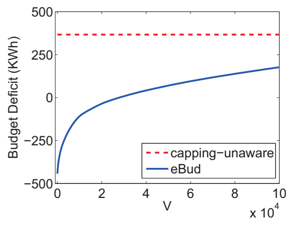


(b) Electricity price.

Figure 2. Workload trace and electricity price



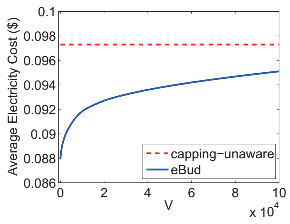
(a) The average cost of eBud approaches the lower bound set by capping-unaware algorithm.



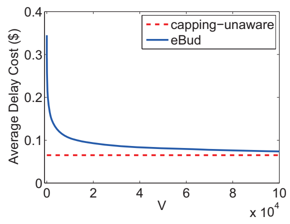
(b) Energy budget deficit increases as V increases. eBud achieves zero budget deficit at $V \approx 2.5 \times 10^4$.

Table I
Normalized server rate and power consumption.

Speed	1.197	1.463	1.862	2.128	2.527
Power Consump. (W)	201	210	233	255	300



(c) Electricity cost increases as V increases.



(d) Delay cost decreases as V increases.

Figure 3. Impact of V.

V. Simulation

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

A. Data Sets

We consider a university data center with 100 physical servers that host a maximum of 400 VMs. Each physical server has a maximum power of 300W, an idle power of 190W, and 5 different speed settings as considered in [24] and shown in Table I. As in the existing work [6], [12], [13], we only model the server power consumption for delay-sensitive workloads without considering delay-tolerant batch jobs. The duration of each time slot is 5 minutes. The budgeting period in our study is one month, and the default total energy budget is 8352KWh (i.e., 96% of the energy consumed by the optimal capping-unaware algorithm without any energy capping). The peak power constraint is set as 25KW and the weighting parameter converting the delay to monetary cost is $\beta = 0.0005$.

• Workloads: We have profiled the server usage log of Florida International University (FIU, a large public university in the U.S. with over 50,000 students) for January, 2012, and add 50% random noise to generate the workload for 400 VMs. We show the workload of one of the 400 VMs in Fig. 2(a), normalized with respect to the maximum service rate of 80*2.2527.

• Electricity price: As in [6], [12], we consider that the data center participates in a real-time electricity market and obtain from [25] the hourly electricity price for Mountain View, California, during January, 2012.

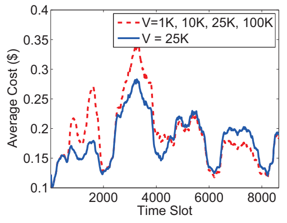
• Others: As aforementioned, we assume the VM placement decision (and hence turning on/off servers, too) is orthogonal to our work. In the simulation, the number of active servers is roughly proportional to the workload, and workloads are distributed such that each server receives (approximately) the same amount of workloads. 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 characteristics: in our simulation, we use 10 different constants of proportionality to simulate 10 different types of jobs served by the data center. In particular, the constants of proportionality are predetermined before the simulation but randomly chosen from the following set: $80\times[0.80, 0.85, 0.90, 0.95, 1.00, 1.05, 1.10, 1.15, 1.20, 1.25]$.

B. Simulation Results

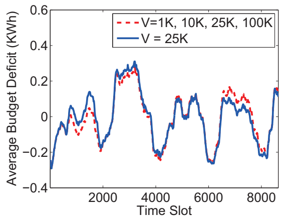
We now present the simulation results based on the above settings.

1) Impact of V: We now show how the value of V affects eBud.

Constant V. We first consider a constant V throughout the budgeting period. Fig. 3(a) and Fig. 3(b) show the impact of V on the average cost per time slot (i.e., $\bar{g}$) and the total energy budget deficit, respectively. The result conforms with our analysis that with a greater V, eBud is less concerned with the energy budget deficit while caring more about the cost. This can also be seen from Fig. 3(c) and Fig. 3(d) that show the average electricity cost and delay cost, respectively: when V increases, the server runs in higher speed, leading to a better delay performance while resulting in more power consumption hence electricity cost. In the extreme case in which V goes to infinity, eBud reduces to a *capping-unaware* algorithm that minimizes the cost without considering energy capping. Clearly, capping-unaware algorithm achieves a cost that is a lower bound

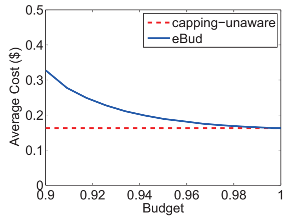


(a) Moving average cost.

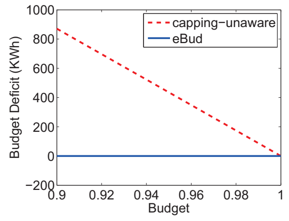


(b) Moving average budget deficit.

Figure 4. Impact of time-varying V.

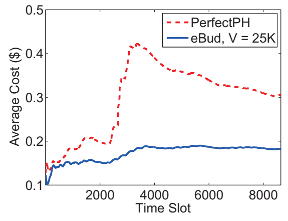


(a) With a reduced budget, eBud incurs a slight increase in the average cost while still satisfying the energy budget.

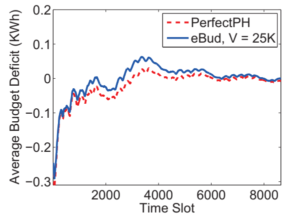


(b) Capping-unaware algorithm incurs a high energy budget deficit, while eBud achieves a zero deficit.

Figure 5. Impact of energy budget.



(a) eBud incurs a lower cost than PerfectPH. A sharp increase in average cost is incurred by PerfectPH when workload spikes.



(b) eBud achieves zeros budget defict at the end while PerfectPH have some unused budget

Figure 6. Comparison with prediction-based methods.

on the cost that can be possibly achieved by any algorithm satisfying the long term energy capping constraint. It can be seen from Fig. 3(a) and Fig. 3(b) that the cost achieved by eBud is fairly close to the lower bound on the cost achieved by the capping-unaware algorithm when V is approximately 2.5×10^4, whereas eBud still satisfies the long term energy capping constraint.

Varying V. We show in Fig. 4 the impact of dynamically changing V over the course of operation. Specifically, we change V every 7.5 days and present the moving average cost and budget deficit per time slot (averaged over the past 48 hours) in Fig. 4(a) and Fig. 4(b), respectively. The fluctuation of moving average values is mainly due to the large variation of workloads over the month. We observe from Fig. 4 that, by choosing a small V initially, the average cost is quite big whereas it can be significantly reduced later by increasing the value of V (at the expense of increasing the energy budget deficit). This indicates the flexibility of dynamically tuning V to adjust the tradeoff between cost minimization and the potential violation of energy capping constraint.

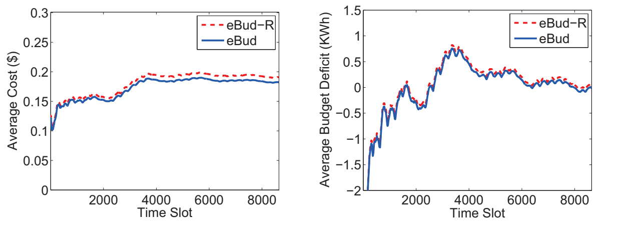
In Figs. 3 and 4, we do not show the optimal offline algorithm with T-step lookahead information, because it cannot possibly achieve a cost less than the optimal capping-unaware algorithm, compared to which eBud already achieves a close-to-minimum cost.

2) Impact of long-term energy budget Z: We now show in Fig. 5 the impact of long-term energy budget Z on the cost and budget deficit. Under our simulation settings, the capping-unaware algorithm consumes 8712 KWh electricity energy over one month, which we normalize to 1. We appropriately choose V such that eBud achieves a zero budget deficit. Naturally, the average cost of eBud increases when the energy budget decreases. However, it can be seen that given a 95% energy budget, eBud only exceeds the capping-unaware algorithm by approximately 10% in terms of the average cost, while still being able to satisfy the energy capping constraint (whereas the capping-unaware algorithm clearly violates the energy budget), as shown in Figs. 5(a) and 5(b). This indicates the flexibility of eBud in satisfying various budget constraints while resulting in a satisfactory cost.

3) Comparison with prediction-based method: This simulation compares eBud with a prediction-based method for energy/cost capping as considered in [3]–[5], the best known existing solution to resource management with a long-term energy/cost capping constraint. While there exist other online resource management solutions, none of them have considered the long-term energy budget and thus are excluded from the performance comparison with eBud. Next, we incorporate the nonlinear delay function to the existing prediction-based method [3], [4] and consider a heuristic variation as follows.

- Perfect Prediction Heuristic (**PerfectPH**): The data center operator leverages perfect prediction of 5-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 5-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. 6 shows the comparison between eBud and the prediction-based PerfectPH in terms of the average cost and budget deficit per time slot. Fig. 6(a) demonstrates that eBud is more cost-effective compared to the prediction-based methods with a cost saving of more than 60% over the budgeting period of one month, while both algorithms meet the long term budget constraint. The cost saving



(a) Negligible impact on average cost. (b) Negligible impact on meeting long-term budget.

Figure 7. Robustness against under-estimation of VM service rates.

mainly comes from the fact that eBud can focus on cost minimization even though the workload spikes and the energy budget is *temporarily* violated. By contrast, without foreseeing the long-term future, short-term prediction-based PerfectPH may over-allocate the energy budget at inappropriate time slots and thus have to set a stringent budget for certain time slots when the workload is high, thereby significantly increasing the delay cost. In Fig. 6(b), the average budget deficit is shown over the budgeting period. The lower average budget deficit by PerfectPH is because for some time slots, PerfectPH assigns more budget than the maximum possible power consumption with all the active servers (decided by VM placement controller) running at their maximum speeds (and hence maximum power). 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. 3(a) already demonstrates that eBud is fairly close to the lower bound on the cost while satisfying the budget constraint. This implies that only using 5-minute-ahead prediction in eBud is sufficiently good in terms of cost minimization.

4) Sensitivity study: In practice, it is quite challenging to model accurately the service rate given a certain CPU resource allocation [14], [15]. To cope with this practical challenge and avoid VM overloading, we adopt a conservative approach: we randomly *under-estimate* the service rate by $10-20\%$ given a CPU resource allocation for each type of workload. Mathematically, the *estimated* proportionality factor that relates the CPU resource allocation to the service rate is decreased by $10-20\%$, while the actual service rate is still used when calculating the actual electricity and delay cost. We refer to this variant of eBud as eBud-R. For both eBud and eBud-R, we choose the same cost-capping parameter $V = 2.5 \times 10^4$. It can be seen from Fig. 7(a) and Fig. 7(b) that the performance of eBud-R is quite close to that of eBud, demonstrating that eBud can be successfully applied even though the service rates are conservatively under-estimated.

Other sensitivity studies are also performed, demonstrating that eBud provides a satisfactory performance given various workloads and even with 5-minute-ahead workload prediction errors. These results are omitted due to space limitations.

VI. Related Work

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

Data center optimization and VM resource allocation. There has been a growing interest in optimizing data center operation from various perspectives such as cutting electricity bills [6], [12], [17], [26] and minimizing response times [8], [13]. 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 [26]. As data centers are becoming increasingly virtualized, VM resource management has attracted much research interest: e.g., [27] studies energy-efficient load balancing for web services; [10] proposes admission control and dynamic CPU resource allocation to minimize the cost while bounding the queueing delay for batch jobs; [18]–[20] 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) [8], optimally allocating the limited power budget to servers is crucial for performance improvement. In [8], the peak power budget is optimally allocated to (homogeneous) servers to minimize the total response time based on a queueing-theoretic model; [7] 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., [3], [5], rely on long-term prediction of the future information, which may not be feasible in practice. Similarly, [4] 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 [3], 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. Our prior work [28] studies energy budgeting for a data center with intermittent on-site renewable energy supplies, but it does not consider virtualized systems. To our best knowledge, energy budgeting for virtualized data centers has not been studied by any prior work.

VII. Conclusion

In this paper, we studied energy budgeting for virtualized data centers and proposed an online algorithm, eBud, which determines the server CPU speed and VM resource allocation decision 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 also performed a trace-based simulation study to complement the analysis. The results show that eBud reduces the cost by more than 60% (compared to state-of-the-art prediction-based method) while resulting in the same energy consumption.

Acknowledgment

This work is supported in part by NSF under projects CNS-0969013, CNS-0917021, and CNS-1018108.

References

[1] Google, "Google's green ppas: What, how, and why."

[2] Microsoft, "Becoming carbon neutral: How microsoft is striving to become leaner, greener, and more accountable."

[3] K. Le, R. Bianchini, T. D. Nguyen, O. Bilgir, and M. Martonosi, "Capping the brown energy consumption of internet services at low cost," in *IGCC*, 2010.

[4] Y. Zhang, Y. Wang, and X. Wang, "Electricity bill capping for cloud-scale data centers that impact the power markets," in *ICPP*, 2012.

[5] C. Ren, D. Wang, B. Urgaonkar, and A. Sivasubramaniam, "Carbon-aware energy capacity planning for datacenters," in *MASCOTS*, 2012.

[6] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *SIGMETRICS*, 2011.

[7] H. Lim, A. Kansal, and J. Liu, "Power budgeting for virtualized data centers," in *USENIX ATC*, 2011.

[8] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *SIGMETRICS*, 2009.

[9] M. J. Neely, *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[10] R. Urgaonkar, U. C. Kozat, K. Igarashi, and M. J. Neely, "Dynamic resource allocation and power management in virtualized data centers," in *IEEE/IFIP NOMS*, 2010.

[11] S. Ghiasi, T. Keller, and F. Rawson, "Scheduling for heterogeneous processors in server systems," in *Computing Frontiers*, 2005.

[12] 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 *IEEE Infocom*, 2010.

[13] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *IGCC*, 2012.

[14] S. Kundu, R. Rangaswami, A. Gulati, K. Dutta, and M. Zhao, "Modeling virtualized applications using machine learning techniques," in *VEE*, 2012.

[15] L. Wang, J. Xu, and M. Zhao, "Modeling vm performance interference with fuzzy mimo model," in *Feedback Computing*, 2012.

[16] S. Liu, S. Ren, G. Quan, M. Zhao, and S.-P. Ren, "Profit-aware load balancing for distributed cloud data centers," in *IEEE/IFIP IPDPS*, 2013.

[17] N. U. Prabhu, *Foundations of Queueing Theory*. Kluwer Academic Publishers, 1997.

[18] V. Shrivastava, P. Zerfos, K.-W. Lee, H. Jamjoom, Y.-H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Infocom*, 2011.

[19] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *GREENCOM-CPSCOM*, 2010.

[20] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *HPDC*, 2011.

[21] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *USENIX ATC*, 2009.

[22] 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.

[23] M. A. Islam, S. Ren, and G. Quan, "Online energy budgeting for virtualized data centers," *Supplemenary materials*, Available at: http://users.cis.fiu.edu/~sren/doc/tech/mascots_2013_full.pdf.

[24] X. Chen, X. Liu, S. Wang, and X.-W. Chang, "Tailcon: Power-minimizing tail percentile control of response time in server clusters," in *SRDS*, 2012.

[25] California ISO, "http://www.caiso.com/."

[26] B. Guenter, N. Jain, and C. Williams, "Managing cost, performance and reliability tradeoffs for energy-aware server provisioning," in *IEEE Infocom*, 2011.

[27] C.-T. Yang, K.-C. Wang, H.-Y. Cheng, C.-T. Kuo, and W. C. C. Chu, "Green power management with dynamic resource allocation for cloud virtual machines," in *HPCC*, Washington, DC, USA, 2011.

[28] A. H. Mahmud and S. Ren, "Online resource management for data center with energy capping," in *Feedback Computing Workshop*, 2013.