COCA: Online Distributed Resource Management for Cost Minimization and Carbon Neutrality in Data Centers

Shaolei Ren
Florida International University
sren@cs.fiu.edu

Yuxiong He
Microsoft Research, Redmond
yuxhe@microsoft.com

ABSTRACT

Due to the enormous energy consumption and associated environmental concerns, data centers have been increasingly pressured to reduce long-term net carbon footprint to zero, i.e., *carbon neutrality*. In this paper, we propose an online algorithm, called COCA (optimizing for COst minimization and CArbon neutrality), for minimizing data center operational cost while satisfying carbon neutrality without long-term future information. Unlike the existing research, COCA enables distributed server-level resource management: each server autonomously adjusts its processing speed and optimally decides the amount of workloads to process. We prove that COCA achieves a close-to-minimum operational cost (incorporating both electricity and delay costs) compared to the optimal algorithm with future information, while bounding the potential violation of carbon neutrality. We also perform trace-based simulation studies to complement the analysis, and the results show that COCA reduces cost by more than 25% (compared to state of the art) while resulting in a smaller carbon footprint.

Keywords

Carbon neutrality, Data center, Load distribution, Resource management, Scheduling, Stochastic control.

1. INTRODUCTION

To satisfy the exploding demand of Internet and cloud computing services, large-scale data centers consisting of tens of thousands of servers require many megawatts of electricity power, a significant portion of which comes from coal or other carbon-intensive sources that produce huge carbon footprints [2, 29]. In view of the growing trend of data center energy consumption and the associated environmental concerns, large IT companies have been increasingly pressured to reduce their net carbon footprint to zero (i.e., achieve *carbon neutrality* or net-zero) for sustainable computing, either mandated by governments in the form of Kyoto-style protocols, required by utility companies, voluntarily, or urged by non-profit environmental organizations [3, 14, 17, 25]. In addition to highly-desired sustainability, carbon neutrality also enables other remarkable benefits for data centers such as tax reduction, favorable accreditation, lower-cost contracts with power utility companies, and/or business promotion [3, 14, 25, 31]. Recently, several companies such as Google, Microsoft and Facebook have set carbon neutrality as their long-term strategic goals [3, 14, 25].

Using green energy directly from on-site projects or utility companies might seem to be an intuitive way to achieve carbon neutrality, but it is not yet widely practical. In general, the best location for building a data center may not be the most desired location for generating sufficient green energies to satisfy the data center requirement. Moreover, as recently pointed out by [15], directly drawing green-source electricity energy from utility companies (at higher prices, called "renewable energy tariffs") to power data centers is still under debates and subject to state-level approvals as well as multiple practical challenges (e.g., large-scale cost-effective renewable resources). Thus, to accomplish carbon neutrality, data centers normally rely on a bundle of approaches, such as generating on-/off-site green (or renewable) energy and purchasing renewable energy credits (RECs), to offset brown energy (or electricity) usage.

Achieving carbon neutrality is desirable yet challenging: it needs to budget data centers' electricity usage over a long timescale such that the "unknown" future brown energy consumption can be completely offset by limited renewables. In other words, carefully "budgeting the energy usage" (which we refer to as *energy budgeting*) over a long term is crucial for being carbon neutral, and meanwhile neither operational costs nor quality of service can be considerably compromised. While the existing *power* budgeting technique that allocates the peak power to severs given the current workload (e.g., [18]) might seem to be applicable for our purpose, energy budgeting fundamentally differs from power budgeting and faces a significant challenge: data center operator needs to decide its energy usage in an online manner that cannot possibly foresee the far future time-varying workloads or intermittent renewable energy availability. Although some preliminary efforts have been made for a related problem of energy usage capping in data centers [8, 17], they require accurate prediction of future information (e.g., workloads, green energy availability) that is often unavailable in practice, especially considering the unforeseeable traffic spikes [10, 31] and intermittent supply of solar and wind energies heavily

SC'13 November 17-21, 2013, Denver, CO, USA

http://dx.doi.org/10.1145/2503210.2503248

depending on weather conditions.

In this paper, we study energy budgeting for carbon neutrality and propose a provably-efficient online algorithm, called COCA (optimizing for COst minimization and CArbon neutrality), for minimizing the data center *operational* cost (incorporating both electricity and delay costs) while satisfying carbon neutrality without requiring long-term future information. In stark contrast with the recent research [24] that coarsely studies data center-level resource management (i.e., controlling the number of active homogeneous servers) for long-term energy capping, COCA explores finer-grained server-level resource management, incorporates load distribution and enables distributed implementation in a data center with heterogeneous servers. Specifically, by using COCA, each server can autonomously adjust its processing speed (and hence, power consumption, too) and optimally decide the amount of workloads to process. By extending the recently developed Lyapunov optimization technique [26], we prove that COCA can achieve a close-to-minimum operational cost, compared to the optimal offline algorithm with lookahead information, while bounding the maximum carbon deficit.

We also perform a trace-based simulation study to validate the analysis. The results show that **(a)** COCA achieves a close-to-minimum cost while still satisfying carbon neutrality; **(b)** COCA reduces operational cost by more than 25% compared to the state-of-the-art prediction-based method while more accurately satisfying the desired carbon neutrality; and **(c)** COCA is robust against various factors, demonstrating a significant potential for practical applicability.

In summary, this paper rigorously addresses carbon neutrality in the absence of long-term future information. Compared to the existing studies, our work significantly differs in the following aspects.

• Objective: Our work minimizes the operational cost while satisfying the increasingly important long-term carbon neutrality, differing from most of the existing research that directly minimizes the instantaneous operational cost (e.g., [19, 21, 30]) but typically does not guarantee carbon neutrality.

• Approach: COCA adopts an *online* approach to achieve carbon neutrality, whereas the existing relevant studies on energy capping (but not explicitly on carbon neutrality, e.g., [17]) requires prediction of long-term future information that may not be practically available. Moreover, unlike the recent research [23, 24] that only considers data center-level resource management (i.e., number of on/off homogeneous servers) in a centralized manner, COCA explores distributed server-level resource management and load distribution to achieve carbon neutrality.

• Performance: Unlike the existing prediction-based heuristic approaches [17,21], COCA offers analytical bounds on the operational cost and potential deviation from carbon neutrality, compared to the optimal offline algorithm with lookahead information. The effectiveness of COCA in practical settings is also validated through simulation studies.

The rest of this paper is organized as follows. Section 2 describes the model. In Section 3, we present the problem formulation, and in Section 4, we develop our online algorithm, COCA, as well as provide its performance analysis. The analysis is further validated by a simulation study in Section 5. Related work is reviewed in Section 6 and finally, concluding remarks are offered in Section 7.

Table 1: List of key notations.

Notation	Description
$x_i(t)$	Service rate of server i
$\lambda_i(t)$	Workloads processed by server i
$p(t)$	Server power consumption
$r(t)$	On-site renewable energy
$f(t)$	Off-site renewable energy
Z	Total RECs
$e(t)$	Electricity cost
$d(t)$	Delay cost
$g(t)$	Total cost
V	Cost-carbon parameter
$q(t)$	Carbon deficit queue length

2. MODEL

We consider a discrete-time model by dividing the entire budgeting period (e.g., typically a year) into J time slots, each of which has a duration that matches the timescale of prediction window for which the data center operator can *accurately* predict the future information (including the workload arrival rate, renewable energy supply, and electricity price). In the following analysis, we mainly focus on hour-ahead prediction, which is sufficiently good in terms of cost minimization as shown in Section 5. Throughout the paper, we also use *environment* to collectively refer to electricity price, on-site/off-site renewable energy supplies and workloads. Next, we present the modeling details. Key notations are summarized in Table 1.

2.1 Data center

We consider one data center that has N servers, and in general, these servers may be heterogeneous in their power consumption and processing speeds due to various reasons such as different purchase dates. Moreover, each server may trade performance for power consumption by varying its performance and power states (e.g., P-states, C-states, or a combination of them) or varying its processing speeds (e.g., via dynamic voltage and frequency scaling or DVFS [22]). Note that the processing speed is quantified in terms of the *service rate*, i.e., how many jobs can be processed in a unit time on average. We interchangeably use "processing speeds" and "service rates" wherever applicable. We also use "capacity" to represent the service rate of a server. To keep our model general, we consider that server i can choose its speed x_i out of a finite set $\mathcal{S}_i = \{s_{i,0}, s_{i,1}, \cdots, s_{i,K_i}\}$, where $s_{i,0} = 0$ represents zero speed (e.g., deep sleep or shut down) and K_i is the total number of positive processing speeds available to server i. Next, assuming that the servers consume a negligible power under the zero-speed mode, we express the average power consumption of server i as

$$p_i(\lambda_i, x_i) = \begin{cases} p_{i,s} + p_{i,c}(x_i) \cdot \frac{\lambda_i}{x_i}, & \text{if} x_i > 0, \\ 0, & \text{if} x_i = 0, \end{cases} \quad (1)$$

where λ_i is the workload arrival rate distributed to server i, $p_{i,s}$ is the static power regardless of the workloads as long as server i is turned on, and $p_{i,c}(x_i)$ is the computing power incurred only when server i is processing workloads at a speed of x_i.

In our study, we 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 can be conveniently absorbed by a (time-varying) power usage effectiveness (PUE) factor[1] that, multiplied by the server power consumption, yields the total data center power consumption [21]. Thus, the total power consumption[2] at time t is given by

$$p(\vec{\lambda}(t), \vec{\mathbf{x}}(t)) = \sum_{i=1}^{N} p_i(\lambda_i(t), x_i(t)), \quad (2)$$

where $\vec{\lambda}(t) = (\lambda_1(t), \cdots, \lambda_N(t))$ and $\vec{\mathbf{x}}(t) = (x_1(t), \cdots, x_N(t))$ are the load distribution and capacity provisioning decisions for time t, respectively.

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 [21, 30]). Assuming that the amount of available on-site renewable energy that can be used to process workloads is $r(t)$ (as specified in the next subsection), we can express the incurred electricity cost as

$$e(\vec{\lambda}(t), \vec{\mathbf{x}}(t)) = w(t) \cdot \Big[p(\vec{\lambda}(t), \vec{\mathbf{x}}(t)) - r(t)\Big]^+, \quad (3)$$

where $[\,\cdot\,]^+ = \max\{\cdot, 0\}$ indicating that no electricity will be drawn from the power grid if on-site renewable energy is already sufficient. While we use (3) to represent the electricity cost for the data center at time t (as considered by [21, 30]), 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).

2.2 Renewable energy

We consider three representative types of renewable energy sources that have been increasingly adopted by large data centers in many regions of the world [14, 25].

On-site renewable energy: Renewable energy generators such as solar panels and wind turbines can be easily installed on-site and directly provide green energy to power data centers [14], but they are highly dependent on weather conditions, exhibiting an intermittent nature. We denote the available on-site renewable energy supply during time t by $r(t) \in [0, r_{\max}]$.

Off-site renewable energy: Since the most desirable locations for renewable energy generation and building data centers are typically different, large data centers now resort to off-site renewable energy to achieve carbon neutrality [14, 25]. One important and widely-used type of off-site renewable energy is through power purchasing agreement (PPA). For example, Google has invested in and signed PPAs with several renewable energy plants such that the generated renewable energy will be directly fed into the local electricity grid and then used to offset the brown energy usage of Google's data centers [14]. Nonetheless, data centers still need to draw electricity from the grid and pay utility companies for accountability reasons, since off-site renewable energy becomes undifferentiated with other types of energies once it enters the grid [14, 25]. We denote the available off-site renewable energy generated via PPAs for time t as $f(t) \in [0, f_{\max}]$.

REC: RECs are a tradable commodity in energy markets that may be purchased to offset data centers' brown energy usage, whereas they are not tied to any physical delivery of electricity [14]. While our model accommodates various approaches to purchasing RECs (e.g., dynamic purchase in real time), we assume that a (fixed) amount of RECs, denoted by Z have been purchased prior to a budgeting period.

Before proceeding to the workload model, we note that our consideration of the above three renewable sources is mostly based on the current industry practice that large data centers do not have the option of directly getting green energy from local utility companies [14, 15, 25]. Nonetheless, even though directly purchasing renewable energy from utility companies becomes a reality in the future, our research is still useful in the sense that COCA can minimize the operational cost while *capping* the long-term energy usage: all the analysis still applies by removing the off-site renewable energy from our model and taking the REC parameter Z as the desired total energy cap.

2.3 Workloads

We denote by $\lambda(t) \in [0, \lambda_{\max}]$ the total arrival rate of workloads in the data center during time t, where $\lambda_{\max}$ is the maximum possible arrival rate. As assumed in prior work [7, 17, 21], the value of $\lambda(t)$ is accurately available at the beginning of each time slot t, while our simulation results further demonstrate the robustness of COCA against inaccurate knowledge of workload arrival rates. As in [19, 21, 30], we focus on delay-sensitive interactive workloads, which can take over 50% of data center workloads [21], while isolating delay-tolerant batch workloads that can be handled by maintaining a separate batch job queue as considered by several existing studies [36]. The workloads first arrive at a load distributor before they are distributed to servers for processing. We denote the workload arrival rate distributed to server i at time t by $\lambda_i(t) \geq 0$.

To quantify the overall data center delay performance, we introduce the notion of *delay cost* capturing the delay-induced revenue loss and/or user dis-satisfaction [21]: we model the cost associated with the workload delay at server i by a convex function $d_i(\lambda_i, x_i)$, which is intuitively increasing in λ_i and decreasing in the service rate x_i [19]. As a concrete example, we can model the service process at each server as an M/G/1/PS (Memoryless/General/1/Processor-Sharing) queue and use the average response time (multiplied by the arrival rate) to represent the delay cost. Specifically, the total delay cost at time t can be written as [28]

$$d(\vec{\lambda}(t), \vec{\mathbf{x}}(t)) = \sum_{i=1}^{N} d_i(\lambda_i(t), x_i(t)) = \sum_{i=1}^{N} \frac{\lambda_i(t)}{x_i(t) - \lambda_i(t)}, \quad (4)$$

in which we ignore the network delay cost between the load distributor and servers. Note that the average network delay between the users and the data center can be approximately modeled as a certain (time-varying) variable [21] and added into (4) without affecting our approach of analysis. While

[1] PUE is defined as the ratio of the total power consumption to IT-equipment power consumption.

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

the M/G/1/PS queueing model may not capture the exact response time in practice, it has been widely used as an analytic vehicle to provide a reasonable approximation for the actual service process [9,21,30]. In addition, our analysis is not restricted to the specific delay cost given by (4).

3. PROBLEM FORMULATION

In this section, we first specify the optimization objective and constraints. Then, we present an offline formulation for our problem on capacity provisioning and load distribution. Finally, we introduce an offline algorithm, which serves as a benchmark that we compare COCA with.

3.1 Objective and constraints

The data center considers workload arrival rate, on-site renewable energy and electricity price as inputs, and decides the server speeds and load distributions. It aims at minimizing the long-term operational cost subject to a set of constraints, as specified below.

Objective. We focus on operational costs rather than capital costs (e.g., building data centers) or other non-IT costs (e.g., human resource cost). 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 user experiences and revenues [19, 21]. Our study incorporates both costs by considering a parameterized cost function as follows[3]

$$g(\vec{\lambda}(t),\vec{\mathbf{x}}(t)) = e(\vec{\lambda}(t),\vec{\mathbf{x}}(t)) + \beta \cdot d(\vec{\lambda}(t),\vec{\mathbf{x}}(t)), \quad (5)$$

where $\beta \geq 0$ is the weighting parameter adjusting the importance of delay cost relative to the electricity cost [21]. The optimization objective is to minimize the long-term average cost expressed as

$$\bar{g} = \frac{1}{J}\sum_{t=0}^{J-1} g\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right), \quad (6)$$

where J is the total number of time slots over the entire budgeting period.

Constraints. To avoid server overloading and workload dropping, the load distribution decisions need to satisfy

$$0 \leq \lambda_i(t) \leq \gamma \cdot x_i(t),\ \forall i,t \quad (7)$$

$$\sum_{i=1}^{N} \lambda_i(t) = \lambda(t),\ \forall t \quad (8)$$

where $\gamma \in (0,1)$ is a predetermined parameter that controls the maximum utilization of a server. Naturally, server i can only select one of its supported service rates, i.e.,

$$x_i(t) \in \mathcal{S}_i = \{s_{i,0}, s_{i,1}, \cdots, s_{i,K_i}\},\ \forall i,t. \quad (9)$$

Note that additional constraints, such as peak power and maximum delay cost, can also be incorporated with little impact on our proposed online algorithm.

Next, we specify the long-term carbon neutrality constraint as follows. While carbon neutrality was originally proposed for "net-zero" carbon emissions, we follow the current market practice and say that a data center achieves carbon neutrality as long as its electricity usage is completely *offset* by the off-site renewable energy plus RECs [14, 25], under the implicit **assumptions** that: (1) electricity energy is brown; and (2) renewable energy and RECs are green with a negligible carbon footprint. Mathematically, the data center desires to follow the long-term constraint specified by

$$\frac{1}{J}\sum_{t=0}^{J-1}\left[p\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right) - r(t)\right]^+ \leq \frac{\alpha}{J}\cdot\left[\sum_{t=0}^{J-1} f(t) + Z\right], \quad (10)$$

where $\alpha > 0$ indicates the desired capping of electricity usage relative to the total off-site renewable energy plus RECs. The less α, more aggressive the data center is in achieving carbon neutrality. In particular, $\alpha \in (0,1)$ means that the data center uses less electricity than the allowed budget, and may sell the leftover budget in carbon markets [17]. Rather than deciding α, COCA works with a given value of α.

3.2 Offline problem formulation

This subsection presents an offline problem formulation for capacity provisioning and load distribution as follows.

$$\mathbf{P1}: \quad \min_{\mathcal{A}} \bar{g} = \frac{1}{J}\sum_{t=0}^{J-1} g\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right) \quad (11)$$

$$s.t., \quad \text{constraints } (7),(8),(9),(10), \quad (12)$$

where $\mathcal{A}$ represents a sequence of decisions, i.e., $\vec{\lambda}(t)$ and $\vec{\mathbf{x}}(t)$, for $t = 0, 1, \cdots, J-1$, which we need to optimize. The first major practical challenge that impedes derivation of the optimal solution to **P1** is the lack of future information: optimally solving **P1** requires complete offline information (e.g., workload arrivals, renewable energy supplies) over the entire budgeting period that is very difficult, if not impossible, to accurately predict in advance. Furthermore, **P1** belongs to mixed-integer nonlinear programming and is difficult to solve, even if the long-term future information is accurately known a priori. Thus, these challenges demand an online approach that can efficiently make resource management decisions without foreseeing the far future.

T-step lookahead algorithm. As in [26], we now introduce an offline algorithm with T-step lookahead information as a benchmark. Specifically, we divide the entire budgeting period into $R \geq 1$ frames, each having $T \geq 1$ time slots such that $J = RT$, and present the following problem formulation:

$$\mathbf{P2}: \quad \min_{\vec{\lambda}(t),\vec{\mathbf{x}}(t)} \frac{1}{T}\sum_{t=rT}^{(r+1)T-1} g\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right) \quad (13)$$

$$s.t., \quad \text{constraints } (7),(8),(9), \quad (14)$$

$$\frac{1}{T}\sum_{t=rT}^{(r+1)T-1}[p\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right) - r(t)]^+ \leq \frac{\alpha\cdot f_r}{T}, \quad (15)$$

where $f_r = \sum_{t=rT}^{(r+1)T-1} f(t) + \frac{Z}{R}$ is the total amount of available off-site renewable energy supplies during the r-th frame plus the total RECs evenly distributed over the R frames. Essentially, **P2** defines a *family* of offline algorithms parameterized by look-ahead window size T.

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

Boundedness assumption: The workload arrival rate $\lambda(t)$, electricity price $w(t)$, as well as renewable energy supplies $r(t)$ and $f(t)$ are finite, for $t = 0, 1, \cdots, J-1$.

[3] Although off-site renewable energy supplies are not free, the payment is often subject to PPAs and not fully controlled by data centers [14, 25].

Algorithm 1 COCA

1: Input $\lambda(t)$, $r(t)$, $w(t)$, at the beginning of each time $t = 0, 1, \cdots, J-1$
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: **P3:**
Choose $\vec{\lambda}(t)$ and $\vec{\mathbf{x}}(t)$ subject to (7)(8)(9) to minimize

$$V \cdot g\left(\vec{\lambda}(t), \vec{\mathbf{x}}(t)\right) + q(t) \cdot \left[p\left(\vec{\lambda}(t), \vec{\mathbf{x}}(t)\right) - r(t)\right]^+ \quad (16)$$

6: Update $q(t)$ according to (17).

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

We denote the minimum average cost for the r-th frame by G_r^*, for $r = 0, 1, \cdots, R$, considering all the decisions that satisfy the the constraints (14)(15) and that have perfect information over the frame. Thus, the minimum long-term average cost achieved by the oracle's optimal T-step lookahead algorithm is given by $\frac{1}{R}\sum_{r=0}^{R-1} G_r^*$.

4. ONLINE RESOURCE MANAGEMENT

In this section, we first develop our online algorithm, COCA, and then show that it is efficient with respect to cost minimization compared to the optimal offline algorithm with T-step lookahead information. COCA enables each server to autonomously decide its processing speed and the amount of workloads to process.

4.1 COCA

On top of a high computation complexity due to the involved mixed-integer programming, a significant challenge of directly solving **P1** is that the long-term carbon neutrality constraint couples the data center decisions across different time slots: using more brown energy at the current time will potentially reduce the energy budget available for future uses, and yet the decisions have to be made without foreseeing the future. To address this challenge, we leverage Lyapunov optimization [26] and construct a (virtual) carbon deficit queue to guide the resource management decisions to follow the long-term carbon neutrality constraint. Specifically, assuming $q(0) = 0$, we construct a carbon deficit queue whose dynamics evolves as follows

$$q(t+1) = \left\{q(t) + \left[p\left(\vec{\lambda}(t), \vec{\mathbf{x}}(t)\right) - r(t)\right]^+ - \alpha \cdot f(t) - z\right\}^+, \quad (17)$$

where $q(t)$ is the queue length indicating how far the current electricity usage deviates from the carbon neutrality constraint, and $z = \frac{\alpha}{J} \cdot Z$ is the average RECs per time slot scaled by α. Next, we present COCA in Algorithm 1.

COCA is purely online and requires only the currently available information as the inputs (i.e., $\lambda(t)$, $r(t)$, $w(t)$, but excluding the off-site renewables $f(t)$ because the carbon deficit queue is updated at the end of each time slot after $f(t)$ is realized). We use $V_0, V_1, \cdots, V_{R-1}$ to denote a sequence of positive control parameters (also referred to as cost-carbon parameters) to dynamically adjust the tradeoff between cost minimization and electricity usage over the R frames, each having T time slots. The importance of cost-carbon parameters V will be revisited in Section 4.3. Lines 2-4 reset the carbon deficit queue at the beginning of each frame r, such that the cost-carbon parameter V can be adjusted and the carbon deficit in a new time frame will not be affected by its value resulting from the previous time frame. Line 5 defines an *online* optimization problem **P3** to decide the capacity provisioning $\vec{x}(t)$ and load distribution $\vec{\lambda}(t)$. By considering the additional term $q(t) \cdot \left[p\left(\vec{\lambda}(t), \vec{\mathbf{x}}(t)\right) - r(t)\right]^+$ in **P3**, the data center operator places a higher weight on the electricity usage: the weighting factor for the electricity usage is scaled by V plus the carbon deficit queue length $q(t)$, whereas the weighting factor for delay cost is only scaled by V. As a consequence, when $q(t)$ increases (i.e., the current electricity usage further exceeds the supplied renewable energy and RECs), minimizing the electricity usage is more critical for the data center operator due to the carbon neutrality constraint. Thus, COCA works following the philosophy of "if violate neutrality, then use less electricity", and the carbon deficit queue maintained without foreseeing the future guides the data center decisions towards meeting carbon neutrality, thereby enabling online decisions.

Now, to complete COCA, it remains to solve the optimization problem **P3**, which will be discussed in the next subsection.

Algorithm 2 GSD: Distributed Optimization for **P3**

1: Initialization: servers choose feasible values and set $\vec{\mathbf{x}}^*(t) \leftarrow \vec{\mathbf{x}}^e(t)$, $\vec{\lambda}^*(t) \leftarrow \vec{\lambda}^e(t)$, $\tilde{g}^* \leftarrow \infty$
2: **if** $\lambda(t) \le \gamma \cdot \sum_{i=1}^N x_i^e(t)$ **then**
3: Obtain $\vec{\lambda}^e(t)$ by minimizing over $\vec{\lambda}(t)$

$$Vg\left(\vec{\lambda}(t), \vec{\mathbf{x}}^e(t)\right) + q(t)\left[p\left(\vec{\lambda}(t), \vec{\mathbf{x}}^e(t)\right) - r(t)\right]^+, \quad (18)$$

subject to (7)(8), and set $\tilde{g}^e$ to the minimum value of (18)
4: $u \leftarrow \frac{\exp\left(\frac{\delta}{\tilde{g}^e}\right)}{\exp\left(\frac{\delta}{\tilde{g}^*}\right)+\exp\left(\frac{\delta}{\tilde{g}^e}\right)}$
5: With a probability of u: servers set $\vec{\mathbf{x}}^*(t) \leftarrow \vec{\mathbf{x}}^e(t)$, $\vec{\lambda}^*(t) \leftarrow \vec{\lambda}^e(t)$ and $\tilde{g}^* \leftarrow \tilde{g}^e$; with a probability of $1-u$: servers set $\vec{\mathbf{x}}^e(t) \leftarrow \vec{\mathbf{x}}^*(t)$
6: **end if**
7: Randomly select a server i; server i randomly selects a processing speed $x_i'(t) \in \mathcal{S}_i$ and sets $x_i(t)^e \leftarrow x_i'(t)$
8: Return $\vec{\mathbf{x}}^*(t)$ and $\vec{\lambda}^*(t)$ if the stopping criterion is satisfied; otherwise, go to Line 3

4.2 Distributed optimization

As aforementioned, **P3** is mixed-integer nonlinear programming. While there exist various centralized techniques (such as Generalized Benders Decomposition [12]) to solve it, distributed solutions are desired such that each server can make autonomous decisions. In this paper, we present a distributed algorithm, called GSD (Gibbs Sampling-based Distributed optimization), based on a variation of Gibbs sampling presented in Algorithm 2.

GSD is a distributed algorithm working as follows: at each iteration, a randomly selected server first autonomously

updates its speed, and then the servers decide their optimal load distribution decisions (also distributedly), after which the servers communicate decisions to each other. Alternatively, a coordinating node may facilitate message passing by collecting and distributing information exchanges, while in this case GSD becomes *semi*-distributed. Line 3 in GSD, i.e., minimizing (18), can be solved efficiently using any distributed optimization techniques (see [5] for a solution based on dual decomposition). Note that during the iterations, servers do not need to actually adjust their speeds or load distribution decisions, which is only needed after the completion of the algorithm. In line 7, to randomly select a server, we can assign each server with a random timer to "compete" for the updating opportunity, like in random channel access in wireless networks. In the event of server failures, only functioning servers need to participate in GSD, while those failed servers do not intervene the execution of the algorithm.

It is known that always choosing a better decision (i.e., greedy approach) may lead to an arbitrarily bad outcome for combinatorial optimization [33,34]. To avoid the inefficiency, GSD explores new solutions by deliberately introducing randomness to decision making (i.e., line 5), even though they may be worse than the current solution. More specifically, each server i, for $i = 1, 2, \cdots, N$, maintains $x_i^*(t)$ as its current processing speed, while exploring (possibly) new processing speed $x_i^e(t)$ to avoid being trapped in a locally optimal solution. We use $\vec{\lambda}^*(t)$ and $\vec{\lambda}^e(t)$ to denote the optimal load distribution decisions corresponding to $x_i^*(t)$ and $x_i^e(t)$, respectively. The parameter $\delta > 0$, referred to as *temperature* [33], is used to control exploration versus exploitation (i.e., the degree of randomness). On one hand, as δ becomes large, GSD is more *greedy* and keeps a new solution with a greater probability if it is better than the current solution (i.e., $\tilde{g}^e \leq \tilde{g}^*$). In this case, however, it takes more iterations to identify the globally optimal solution because GSD may be stuck in a local optimal solution for a long time before successfully exploring other less greedy solutions that lead more efficient outcomes. On the other hand, as $\delta \to 0$, GSD tries to explore all the possible solutions from time to time without convergence, even though they are worse than the current one. Note that the randomness introduced in line 5 is a variation of Gibbs sampling [33]: in line 5, only the old decision and the randomly selected new decision are *sampled* probabilistically, unlike the original Gibbs sampling technique that samples all the possible decisions and hence requires the servers to know the costs for all the possible decisions. Next, we formally prove the optimality of GSD.

Accuracy. Theorem 1 shows that GSD can solve the optimization problem **P3** with an arbitrarily high probability as the temperature $\delta \to \infty$.

THEOREM 1. *As $\delta > 0$ increases, GSD converges with a higher probability to the globally optimal solution that minimizes* (16) *subject to* (7)(8)(9). *When $\delta \to \infty$, Algorithm 2 converges to the globally optimal solution with a probability of 1.*

PROOF. The proof is available in Appendix A. ■

Theorem 1 indicates that using Algorithm 2, the servers can select the optimal processing speeds distributedly with an arbitrarily high probability. To avoid being trapped in a locally optimal solution for a long time and yet achieve a good performance, an advisory approach used in practice is selecting the smoothing parameter δ *adaptively*: a small δ is initially chosen to explore all possible decisions, whereas δ is increased over the iterations such that the servers progressively *concentrate* on better solutions [34].

Complexity. It is worth pointing out that despite being distributed, GSD still incurs a worst-case complexity that is exponential in the number of servers (although in practice a reasonably good solution is often quickly identified). In practice, the computational complexity of GSD can be effectively reduced by making capacity provisioning decisions on a *group* basis: changing speed selections for a whole group of (homogeneous) servers in batch. Our numerical results show that with 200 groups of servers, GSD converges to a reasonably good solution within 1 second. Finally, notice that solving **P3** is *not* restricted to using the presented GSD. Instead, other alternative algorithms can also be applied. Compared to solving the computationally prohibitive *offline* problem **P1** that involves a large number of mixed-integer nonlinear problems coupled by the long-term carbon neutrality constraint, COCA is much more practically realizable because the resource management decision is only made once every time slot and the total complexity associated with making a long sequence of decisions is *amortized* over each time slot.

4.3 Performance analysis

Building upon Lyapunov optimization [26], this subsection presents the performance analysis of COCA in Theorem 2.

THEOREM 2. *Suppose that boundedness and feasibility assumptions are satsified. Then, for any $T \in \mathbb{Z}^+$ and $R \in \mathbb{Z}^+$ such that $J = RT$, the following statements hold.*

a. *The carbon neutrality constraint is* approximately *satisfied with a bounded deviation:*

$$\frac{1}{J}\sum_{t=0}^{J-1}\left[p\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right)-r(t)\right]^+ \leq \frac{\alpha}{J}\cdot\left[\sum_{t=0}^{J-1}f(t)+Z\right]+\frac{\sum_{r=0}^{R-1}\sqrt{C(T)+V_r\left(G_r^*-g_{\min}\right)}}{R\sqrt{T}}, \tag{19}$$

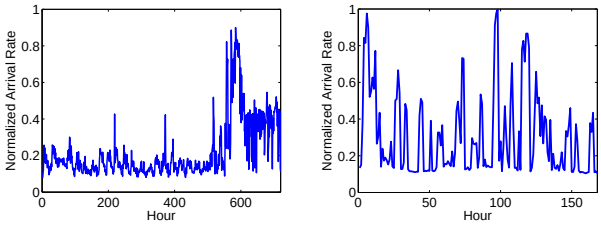
where $C(T) = B + D(T-1)$ with B and D being finite constants, G_r^* is the minimum average cost achieved over the r-th frame by the optimal offline algorithm with T-slot lookahead information, for $r = 0, 1, \cdots, R-1$, and $g_{\min}$ is the minimum hourly cost that can be achieved by any feasible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

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

$$\bar{g}^* \leq \frac{1}{R}\sum_{r=0}^{R-1}G_r^* + \frac{C(T)}{R}\cdot\sum_{r=0}^{R-1}\frac{1}{V_r}. \tag{20}$$

PROOF. The proof is available Appendix B. ■

Theorem 2 shows that, given a fixed value of T and R, COCA is $O(1/V)$-optimal with respect to the average cost against the optimal T-step lookahead policy, while the carbon neutrality constraint is guaranteed to be *approximately* satisfied with a bounded "fudge factor" of $\frac{\sum_{r=0}^{R-1}\sqrt{C(T)+V_r\left(G_r^*-g_{\min}\right)}}{R\sqrt{T}}$. Approximate satisfaction of carbon neutrality stems from the fact that Theorem 2 applies for an arbitrarily changing environment satisfying



(a) FIU workloads. (b) MSR workloads.

Figure 1: Workload trace.

boundness and feasibility assumptions: workloads cannot be dropped and it cannot be ruled out that workloads are constantly very high during the last few time slots of the budgeting period, thereby deviating from carbon neutrality even though neutrality is satisfied for the prior time slots. In practice, data centers may purchase additional RECs at the end of a budgeting period to offset the remaining electricity usage.

***Dynamic* selection of cost-carbon parameters.** Supporting dynamic selection of cost-carbon parameters is important because the appropriate value of V depends on specific modeling parameters and is typically determined on a trial-and-error basis. For example, if the current cost is too high whereas the electricity usage is far below the allowed budget, the data center operator can increase the value of V to weaken the carbon neutrality constraint. Nonetheless, the entire budgeting period is limited (e.g., 6 months or a year) and consequently, analyzing the performance of an algorithm under a *constant* value of V, which is difficult to find appropriately in advance, limits its effectiveness to meet the desired cost and energy budget within a finite budgeting period. Therefore, it is important to derive the performance bounds of COCA by explicitly considering time-varying values of $V_0, V_1, \cdots, V_{R-1}$ that are used in practical scenarios. COCA supports adaptation of V by resetting the budget deficit queue at the beginning of each frame consisting of T time slots, effectively decoupling the online decisions across different frames.

5. SIMULATION

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

• The impact of V: We show how cost minimization and satisfaction of carbon neutrality varies with different values of V.

• Comparison with prediction-based method: We compare COCA with the state-of-art prediction-based method and show that COCA reduces the average cost by more than 25% while satisfying the desired carbon neutrality better.

• Sensitivity study: We demonstrate that COCA is robust against several factors such as inaccurate knowledge of the current workload arrival rates.

5.1 Data sets

We consider a data center with a peak server power of 50MW (approximately 216K servers in total, each with a maximum power of 231W). As in the existing work [21, 30], we only model the server power consumption for delay-sensitive workloads as our main focus; we do not model the cooling power or server power for delay-tolerant batch jobs. Due to the practical difficulty in implementing COCA in a real system, we adopt event-based simulations with real-world trace data to validate our analysis, which is a common approach in the literature [21, 30].

• Server: We use Powerpack [11] to measure the power consumption of a server with a quad-core AMD Opteron 2380 processor that supports four different speeds via DVFS. Specifically, if turned on, each server has an idle power of 140W and supports the following four different processing speeds/powers: 0.8GHz (184W), 1.3GHz (194W), 1.8GHz (208W), and 2.5GHz (231W). When running at the maximum speed, we assume each server can process 10 requests per second on average.

• Workloads: We consider "mice-type" synthetic workloads (e.g., web requests), and use real-world trace to drive our simulation. The service time of an individual request follows an exponential distribution with a mean of 100ms (when the server is running at its full speed), which may not perfectly capture a real system but suffices for our evaluation purpose. As the default workload trace, we profile the server I/O usage log of Florida International University (FIU, a large public university in the U.S. with over 50,000 students) from January 1 to December 31, 2012. We show in Fig. 1(a) the trace of July, 2012, normalized with respect to the maximum arrival rate. The trace exhibits a significant increase around late July, 2012, due to the summer activities. We scale the FIU workload trace such that the maximum service request arrival rate is 1.1million req/sec (approximately 50% of the data center capacity when all the servers are running at their full speeds). For sensitivity studies, we also use workload trace for Microsoft Research (MSR) first shown in [19] and repeat the trace for one year by adding random noises of up to ±40%. The I/O trace for MSR is taken from 6 RAID volumes at MSR Cambridge, and the traced period is 1 week starting from 5PM (GMT) on February 22, 2007 [19]. Fig. 1(b) shows the normalized MSR workload trace for one week.

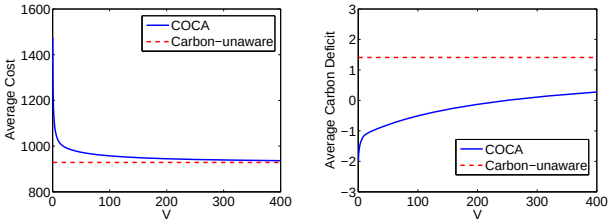
• Renewable energy: We obtain from [1] the hourly renewable energies (generated through solar panels and wind turbines) for the city of Mountain View as well as the state of California during the year of 2012, and scale them proportionally such that on-site renewable accounts for approximately 20% of the total energy consumption.

• Electricity price: As in [21, 29, 30, 35], we assume that the data center participates in a real-time electricity market and obtain from [1] the hourly electricity price for Mountain View.

• Others: The budgeting period in our study is one year, and the default total allowed electricity usage is 92% of 1.55×10^5MWh (i.e., 1.43×10^5MWh), where 1.55×10^5MWh is the electricity usage of carbon-unaware algorithm (to be specified later). Among the 1.43×10^5MWh renewable budget, off-site renewable energy and RECs contribute 40% and 60%, respectively. The weighting parameter converting the delay to monetary cost is $\beta = 10$.

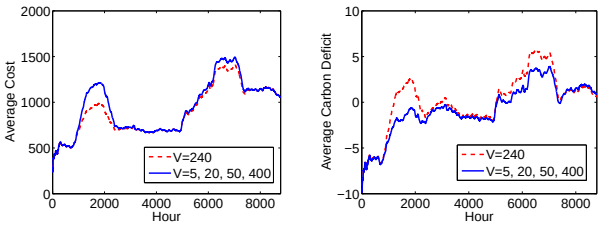
5.2 Results

We drive the event-based simulation using the above synthetic trace data. The power consumption and delay are recorded as outputs of the simulation. Next, we present



(a) Cost decreases as V increases.

(b) Carbon deficit increases as V increases.



(c) Moving average cost given time-varing V.

(d) Moving average carbon deficit given time-varing V.

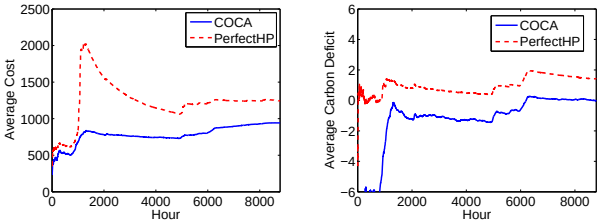
Figure 2: Impact of V.

the simulation results as follows.

5.2.1 *Impact of V*

Constant V. We first consider a constant V throughout the budgeting period (i.e., the year of 2012). Fig. 2(a) and Fig. 2(b) show the impact of V on the average hourly cost (i.e., $\bar{g}$) and average hourly carbon deficit (i.e., average hourly electricity usage minus the available carbon budget consisting of both off-site renewable energy and RECs), respectively. Note that carbon deficit may be either positive or negative, depending on the amount of offiste renewable energies plus RECs: negative deficit means off-site renewable energies plus RECs exceed the electricity usage. The result conforms with our analysis that with a greater V, COCA is less concerned with the carbon deficit while caring more about the cost. In the extreme case in which V goes to infinity, COCA reduces to a *carbon-unaware* algorithm that minimizes the cost without considering carbon neutrality. Compared to the carbon-unaware algorithm, COCA achieves a close-to-minimum cost with $V \approx 240$ while satisfying carbon neutrality, i.e., using 92% of the electricity consumption of the carbon-unaware algorithm.

Varying V. We change V quarterly and present the moving average hourly cost and carbon deficit (averaged over the past 45 days) in Fig. 2(c) and Fig. 2(d), respectively. We choose moving average over a period of 45 days, because it is sufficiently long to show the general trend of cost/carbon deficit. The fluctuation of moving average values is mainly due to the large variation of workloads over the year. We observe that compared to a constant V, 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 carbon deficit). This indicates the flexibility of dynamically tuning V to adjust the tradeoff between cost minimization and potential violation of carbon neutrality.



(a) Average cost.

(b) Average carbon deficit.

Figure 3: Comparing COCA with PerfectHP.

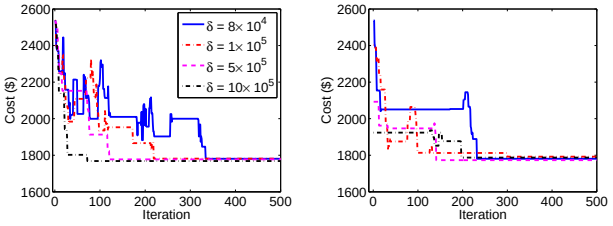
5.2.2 *Comparison with prediction-based method*

This subsection compares COCA with the best known relevant solution — prediction-based method studied in [17, 31]. However, since none of the existing prediction-based methods have considered both the nonlinear delay cost and intermittent off-site renewable energy supplies, we incorporate these factors by considering a heuristic variation as follows.

Perfect hourly prediction heuristic (PerfectHP): The data center operator leverages 48-hour-ahead prediction of hourly workloads and allocates the carbon budget (RECs plus off-site renewables, but not including the on-site renewables) in proportion to the hourly workloads.The operator minimizes the cost subject to the allocated hourly carbon budget; if no feasible solution exists for a particular hour (e.g., workload bust), the operator will minimize the cost without considering the hourly carbon budget. We consider 48-hour-ahead prediction in the comparison, because prediction beyond 48 hours will typically exhibit large errors [21], especially for solar and wind energy supplies that are commonly used for data centers but highly subject to weather condition.

Fig. 3 shows the comparison between COCA and PerfectHP in terms of the average hourly cost and carbon deficit.[4] Fig. 3 demonstrates that COCA is more cost-effective compared to the prediction-based method with a cost saving of more than 25% over one year. COCA achieves the benefits because it can still focus on cost minimization even though the workload spikes and carbon neutrality is *temporarily* violated, since the carbon deficit queue only penalizes the data center for overusing electricity in later time slots while guaranteeing a bounded deviation from the carbon neutrality. By contrast, without foreseeing the long-term future, short-term prediction-based PerfectHP may over-allocate the carbon 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. Note that if 48-hour-ahead prediction information is available, COCA can also leverage it to reduce the cost further, but the potential cost saving is quite limited, because Fig. 2(a) already demonstrates that COCA is fairly close to the lower bound on the cost while satisfying carbon neutrality. This implies that only using hour-ahead prediction in COCA is sufficiently good in terms of cost minimization. In addition to cost saving, COCA also satisfies the desired carbon neutrality constraint better, as

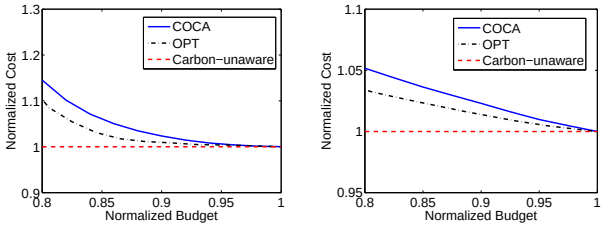
[4]The average at time t in Fig. 3 is obtained by summing up all the values from time 0 to time t and then dividing the sum by $t+1$.



(a) Iteration of total cost with different δ.

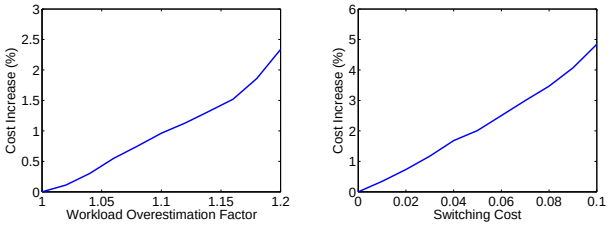
(b) Iteration of total cost with different initial points.

Figure 4: Execution of GSD.



(a) Average cost given different carbon budgets.

(b) Average cost given different carbon budgets for MSR workloads.



(c) Cost increase with workload overestimation.

(d) Cost increase with switching cost.

Figure 5: Sensitivity study.

shown in Fig. 3(b).

5.2.3 *Execution of GSD*

We present a snapshot of executing GSD during the 1500-th time slot (but without considering the queue length) in Fig. 4. Fig. 4(a) shows the total cost given different smoothing parameters δ and matches our analysis: with a larger δ, GSD achieves the minimum cost with a higher probability (while potentially being trapped in a bad solution for a long time). Fig. 4(b) shows the iteration of total cost with different initial points given a fixed $\delta = 10 \times 10^5$. It demonstrates that upon convergence, the proposed distribution algorithm GSD leads to almost the same cost regardless of initial points: GSD is quite insensitive to the initial point. Finally, we note that, to run GSD for 200 groups of servers, the execution time for 500 iterations in our simulator is less than 1 second on a personal desktop computer.

5.2.4 *Sensitivity study*

We perform the following sensitivity studies. For all the cases, we appropriately choose V such that carbon neutrality is satisfied.

• Carbon budget: We now show in Fig. 5(a) the impact of carbon budget (i.e., off-renewables plus RECs) on the cost. Under our simulation settings, the carbon-unaware algorithm consumes 1.55×10^5MWh electricity energy over a year, which we normalize to 1. As a comparison, we also show the optimal offline algorithm (OPT), which has the complete offline information and minimizes the operational cost under carbon neutrality. It can be seen that given a 85% carbon budget — equivalent to using only 85% of electricity, COCA only exceeds the capping-unaware algorithm by approximately 5% in terms of the average cost, while still being able to satisfy carbon neutrality (which is clearly violated by the capping-unaware algorithm). Moreover, COCA works remarkably well even compared to OPT. This demonstrates that, with careful energy budgeting, the long-term energy consumption can be significantly reduced while still keeping the operational cost low (even compared to the optimal offline algorithm and the carbon-unaware algorithm). Note that if a higher carbon budget (e.g., 1.05) is used, COCA will be almost the same as the capping-unaware algorithm without using up the budget, because our optimization objective incorporates both electricity and delay costs and using excessive electricity will increase the total cost (albeit decreasing the delay cost).

• Workload trace: We now consider the MSR workload trace as illustrated in Fig. 1(b). Fig. 5(b) shows the normalized cost achieved by COCA, OPT and the carbon-unaware algorithm under different normalized carbon budgets. It delivers the same message as Fig. 5(a) and demonstrates that COCA works well with different workload traces.

• Workload overestimation: In practice, it may not be possible to perfectly predict hour-ahead workload arrivals. To cope with possible traffic spikes, we can either turn on more servers as a *backup* or directly overestimate the workload arrival rate by a certain overestimation factor $\phi \geq 1$: the higher ϕ, the more overestimates. We choose the later approach, and note that workload overestimation also captures the imperfect modeling of service rates. Fig. 5(c) shows that the total cost only increases by less than 2.5% even when we overestimate the workloads by 20%. This is because although workload overestimation may turn on more servers and incur a higher electricity cost at some time slots, the delay cost will be decreased. Note that we only show the cost with a workload overestimation by up to 20%, because prior research verifies that 20% overestimation is typically sufficient for hour-ahead prediction [20].

• Switching cost: Switching servers on/off induces various costs, such as energy/time waste as well as "wear and tear". As in [19], we incorporate all these factors and use switching cost as the combined cost quantified in terms of *energy* consumption. We normalize the switching cost (incurred by turning on/off one server) with respect to the maximum hourly energy consumption of a single server (i.e., 0.231KWh). Fig. 5(d) shows that even when the switching cost of one server takes 10% of its maximum hourly energy consumption (i.e., 0.0231KWh), the total average operational cost only increases by less than 5%.

We further note that with different combinations of off-site renewables and RECs (but with the same total amount), COCA achieves almost the same cost (less than 1% change), indicating that COCA is not sensitive to renewable energy portfolios, but rather mainly depends on the total budget (as shown in Fig. 5(a)). Other sensitivity studies such as different server settings are also performed, demonstrating

that COCA provides a satisfactory performance in various scenarios and pointing to its applicability in real systems. These results are omitted due to space limitations.

6. RELATED WORK

We now provide a snapshot of the related work.

Data center optimization. There has been a growing interest in optimizing data center operation from various perspectives such as cutting electricity bills [6, 16, 21, 28, 30, 32] and minimizing response times [9, 19]. For example, "power proportionality" via dynamically turning on/off servers based on the workloads has been extensively studied and advocated as a promising approach to reduce the energy cost of data centers [16, 19]. By exploring spatial diversities, [29,32] study geographical load balancing among multiple data centers to minimize energy cost, and [21] proposes to reduce brown energy usage by scheduling workloads to data centers with more green energies. [4, 13, 20] utilize prediction of renewable energy availability for scheduling deferrable batch workloads, thereby minimizing electricity consumption. However, none of these studies have considered carbon neutrality, which is becoming increasingly important for large data center operators [14, 25].

Carbon neutrality and energy capping. The existing studies, e.g., [8,17,31], focus on a related problem of energy capping by using long-term prediction of future information, which may not be feasible in practice. 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 [8,17], their evaluation is empirical only, without providing performance guarantees analytically. In comparison, COCA offers provable guarantees on the average cost while bounding the deviation from carbon neutrality; our simulation results also demonstrate the benefits of COCA over the existing methods empirically. Our recent research [23, 24] studies efficient dynamic server provisioning algorithms for energy capping and carbon neutrality, but load distribution is neglected and only data center-level resource management knob (i.e., controlling the number of active homogeneous servers, or right-sizing the data center) is considered. By contrast, COCA focuses on a practical data center with heterogeneous servers and incorporates server-level resource management (i.e., DVFS) as well as load distribution decisions, enabling a fine-grained management of computing resources in data centers for carbon neutrality. Moreover, COCA features distributed implementation that can be easily scaled to a large system.

7. CONCLUSION

In this paper, we focused on the data center carbon footprint and proposed a provably-efficient online algorithm, called COCA, to control the electricity usage for minimizing the data center operational cost while satisfying carbon neutrality without requiring long-term future information. Unlike prior work, COCA enables a fine-grained and distributed resource management in data center by incorporating server-level CPU speed control and load distribution decisions. Leveraging the recently-developed Lyapunov optimization, we proved that COCA achieves a close-to-minimum operational cost compared to the optimal offline algorithm with lookahead information, while bounding the potential violation of carbon neutrality, in an almost arbitrarily random environment. We also performed a trace-based simulation study to complement the analysis. The simulation results showed that COCA reduces the cost by more than 25% (compared to the state-of-the-art prediction-based method) while resulting in a smaller carbon footprint.

8. REFERENCES

[1] California ISO, `http://www.caiso.com/`.

[2] Electricity from coal, `http://www.powerscorecard.org`.

[3] Facebook statement: Facebook and greenpeace collaboration on clean and renewable energy, `http://www.greenpeace.org`.

[4] B. Aksanli, J. Venkatesh, L. Zhang, and T. Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In *HotPower*, 2011.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] N. Buchbinder, N. Jain, and I. Menache. Online job migration for reducing the electricity bill in the cloud. In *IFIP Networking*, 2011.

[7] N. Deng, C. Stewart, D. Gmach, M. Arlitt, and J. Kelley. Adaptive green hosting. In *ICAC*, 2012.

[8] N. Deng, C. Stewart, D. Gmach, and M. F. Arlitt. Policy and mechanism for carbon-aware cloud applications. In *NOMS*, 2012.

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

[10] 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.*, 30(4):14:1–14:26, Nov. 2012.

[11] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *IEEE Trans. Parallel and Dist. Systems*, 21(5):658–671, May 2010.

[12] A. M. Geoffrion. Generalized benders decomposition. *J. of Optimization Theory and Applications*, 10(4):237–260, 1972.

[13] I. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, and R. Bianchini. Greenslot: Scheduling energy consumption in green datacenters. In *SC*, 2011.

[14] Google. Google's Green PPAs: What, How, and Why, 2011.

[15] Google. Expanding renewable energy options for companies through utility-offered "renewable energy tariffs", 2013.

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

[17] 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.

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

[19] M. Lin, A. Wierman, L. L. H. Andrew, and

E. Thereska. Dynamic right-sizing for power-proportional data centers. In *IEEE Infocom*, 2011.

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

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

[22] J. R. Lorch and A. J. Smith. Pace: A new approach to dynamic voltage scaling. *IEEE Trans. Computers*, 53:856–869, July 2004.

[23] H. Mahmud and S. Ren. Online capacity provisioning for carbon-neutral data center with demand-responsive electricity prices. In *Performance*, 2013.

[24] H. Mahmud and S. Ren. Online resource management for data center with energy capping. In *Feedback Computing*, 2013.

[25] Microsoft. Becoming carbon neutral: How microsoft is striving to become leaner, greener, and more accountable, 2012.

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

[27] D. P. Palomar and M. Chiang. Alternative distributed algorithms for network utility maximization: Framework and applications. *IEEE Trans. Automatic Control*, 52(12):2254–2269, Dec.

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

[29] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. In *SIGCOMM*, 2009.

[30] 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.

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

[32] S. Ren, Y. He, and F. Xu. Provably-efficient job scheduling for energy and fairness in geographically distributed data centers. In *ICDCS*, 2012.

[33] C. Robert and G. Casella. *Monte Carlo Statistical Methods*. New York: Springer-Verlag, 2004.

[34] Y. Song, C. Zhang, and Y. Fang. A game theoretical analysis of joint channel and power allocation in wireless mesh networks. *IEEE J. Sel. Areas Commun.*, 26(7):1149–1159, Sep. 2008.

[35] R. Urgaonkar, B. Urgaonkar, M. J. Neely, and A. Sivasubramaniam. Optimal power cost management using stored energy in data centers. In *SIGMETRICS*, 2011.

[36] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to higher moments of conditional response time. In *SIGMETRICS*, 2005.

APPENDIX

A. PROOF OF THEOREM 1

We note first that once the optimal processing speeds are obtained, the optimal load distribution can be easily derived in a distributed manner (e.g., by using dual decomposition [27]). As a result, to establish Theorem 1, it is equivalent to prove that the servers' processing speeds selected distributedly will converge to the global optimum following the iterations specified by Algorithm 2.

For notational convenience, we drop the time index t wherever applicable and denote the servers' processing speeds by $\vec{\mathbf{x}}$. Following the iterations in Algorithm 2, $\vec{\mathbf{x}}$ evolves as a N-dimension Markov chain in which the i-th dimension corresponds to server i's processing speed decision and has $k_i + 1$ possible values. For the ease of presentation, we begin with a 2-server case and denote the state of the Markov chain as S_{x_1,x_2}, where $x_i \in \mathcal{S}_i = \{s_{i,0}, s_{i,1}, \cdots, s_{i,K_i}\}$ for $i = 1, 2$. As shown in line 7 of Algorithm 2, only one server is selected to explore a new processing speed at each iteration with equal probability among all servers. Thus, we have

$$\begin{aligned}&\Pr(S_{x_1',x_2'} \,|\, S_{x_1,x_2}, \text{server 1 updates})\\ =&\begin{cases}\frac{\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}}')}\right)}{(k_1+1)\left[\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}})}\right)+\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}}')}\right)\right]}, & \text{if } x_1' \neq x_1, x_2' = x_2,\\ 0, & \text{if } x_2' \neq x_2,\\ 1-\sum_{x_1'\neq x_1, x_2'=x_2}\frac{\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}}')}\right)}{(k_1+1)\left[\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}})}\right)+\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}}')}\right)\right]}, & \text{otherwise,}\end{cases}\end{aligned} \tag{21}$$

where $\tilde{g}(\vec{\mathbf{x}}) = V \cdot g\left(\vec{\lambda}, \vec{\mathbf{x}}\right) + q(t)\left[p\left(\vec{\lambda}, \vec{\mathbf{x}}\right) - r(t)\right]^+$, in which $q(t)$ is the carbon deficit queue length at time t and $\vec{\lambda}$ is the optimal load distribution decision given $\vec{\mathbf{x}}$. Thus, the (unconditioned) transition probability can be derived as

$$\begin{aligned}&\Pr(S_{x_1',x_2'} \,|\, S_{x_1,x_2})\\ =&\frac{1}{2}\sum_{i}^{2}\Pr(S_{x_1',x_2'} \,|\, S_{x_1,x_2}, \text{server } i \text{ updates}),\end{aligned} \tag{22}$$

which specifies the probability that the state S_{x_1,x_2} transits to state $S_{x_1',x_2'}$. Furthermore, the Markov chain driven by (22) is irreducible and aperiodic, implying the convergence to a stationary distribution denoted by Ω. Now, we fix the processing speed of one state S_{x_1,x_2} in the Markov chain and derive the following balance equation

$$\begin{aligned}&\sum_{x_2'\neq x_2}\Omega_{S_{x_1,x_2}}\cdot\Pr(S_{x_1,x_2'} \,|\, S_{x_1,x_2})\\ =&\sum_{x_2'\neq x_2}\Omega_{S_{x_1,x_2'}}\cdot\Pr(S_{x_1,x_2} \,|\, S_{x_1,x_2'}).\end{aligned} \tag{23}$$

By plugging (22) into (23) and dividing both sides by $\frac{1}{2}(\frac{1}{k_1+1}+\frac{1}{k_2+1})$, we obtain

$$\begin{aligned}&\sum_{x_2'\neq x_2}\Omega_{S_{x_1,x_2}}\frac{\exp\left(\frac{\delta}{\tilde{g}(x_1,x_2')}\right)}{\exp\left(\frac{\delta}{\tilde{g}(x_1,x_2)}\right)+\exp\left(\frac{\delta}{\tilde{g}(x_1,x_2')}\right)}\\ =&\sum_{x_2'\neq x_2}\Omega_{S_{x_1,x_2'}}\frac{\exp\left(\frac{\delta}{\tilde{g}(x_1,x_2)}\right)}{\exp\left(\frac{\delta}{\tilde{g}(x_1,x_2')}\right)+\exp\left(\frac{\delta}{\tilde{g}(x_1,x_2)}\right)}.\end{aligned} \tag{24}$$

Then, by observing the symmetry of equation (24) and applying the probability conservation law, we obtain the

stationary probability distribution for the Markov chain as

$$\Omega_{\vec{\mathbf{x}}} = \frac{\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}})}\right)}{\sum_{\vec{\mathbf{x}}'}\exp\left(\frac{\delta}{\tilde{g}(\vec{\mathbf{x}}')}\right)}, \tag{25}$$

Denote $\vec{\mathbf{x}}^*$ as the optimal processing speed decision that minimizes $\tilde{g}(\vec{\mathbf{x}})$ over all the possible values of $\vec{\mathbf{x}}$, and suppose temporarily that the optimal decision is unique. By taking the first-order derivative of (25) with respect to δ, it can be seen that the probability of converging to $\vec{\mathbf{x}}^*$ increases with δ. In particular, by letting the temperature $\delta \to \infty$ and noting that $\tilde{g}(\vec{\mathbf{x}})$ is strictly positive for all feasible values of $\vec{\mathbf{x}}$, we obtain

$$\lim_{\delta\to\infty}\Omega_{\vec{\mathbf{x}}} = \begin{cases}1, & \text{if } \vec{\mathbf{x}} = \vec{\mathbf{x}}^*,\\ 0, & \text{otherwise.}\end{cases} \tag{26}$$

Note that if there exist $M \geq 1$ processing speed decisions, all of which minimize $\tilde{g}(\vec{\mathbf{x}})$, then Algorithm 2 will converge to each of the optimal decisions with a probability of $\frac{1}{M}$, as the temperature $\delta \to \infty$.

Finally, we note that the same analysis can be extended to the N-dimensional Markov chain, thereby completing the proof of Theorem 1. ■

B. PROOF OF THEOREM 2

The proof builds upon yet extends the recently-developed Lyapunov optimization technique [26]. Due to space limitations, we only show the key steps. First, we show the following inequality that relates carbon queue length to *approximate* constraint satisfaction:

$$\begin{aligned}&\frac{1}{T}\sum_{t=rT}^{(r+1)T-1}\left[p\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right)-r(t)\right]^+\\ \leq&\frac{\alpha}{T}\left[\sum_{t=rT}^{(r+1)T-1}f(t)+\frac{Z}{R}\right]+\frac{q(rT+T)-q(rT)}{T}.\end{aligned} \tag{27}$$

Next, we define for notational convenience: $y(t) = \left[p\left(\vec{\lambda}(t),\vec{\mathbf{x}}(t)\right)-r(t)\right]^+$, $z(t) = \alpha\cdot f(t)+z$, and $g(t) = g(\vec{\lambda}(t),\vec{\mathbf{x}}(t))$. We also define the quadratic Lyapunov function $L(q(t)) \triangleq \frac{1}{2}q^2(t)$. Let $\triangle_T(t)$ be the $T-$slot Lyapunov drift yielded by some control decisions over the interval $t, t+1, \cdots, t+T-1$: $\triangle_T(t) \triangleq L(q(t+T)) - L(q(t))$. Similarly, the 1-slot drift is $\triangle_1(t) \triangleq L(q(t+1)) - L(q(t))$. Based on $q(t+1) = [q(t)+y(t)-z(t)]^+$, it can be shown that $L(q(t+T)) = \frac{1}{2}q^2(t+1) \leq \frac{1}{2}[q(t)+y(t)-z(t)]^2$. Then, it can be shown that the 1-slot drift satisfies $\triangle_1(t) \leq B + q(t)\cdot[y(t)-z(t)]$, where B is a constant satisfying, for all $t = 0, 1, \cdots, J-1$, $B \geq \frac{1}{2}[y(t)-z(t)]^2$, which is finite due to the boundedness assumption. Next, it can be easily shown

$$\triangle_1(t) + V\cdot g(t) \leq B + V\cdot g(t) + q(t)\cdot[y(t)-z(t)]. \tag{28}$$

The online algorithm described in line 5 of Algorithm 1 actually minimizes the upper bound on the 1-slot Lyapunov drift plus a weighted cost shown on the right hand side of (28). Following (28), we can show that, for $r = 0, 1, \cdots, R-1$, the T-slot drift plus weighted cost satisfies

$$\begin{aligned}&\triangle_T(rT) + V_r\sum_{t=rT}^{rT+T-1}g(t)\\ \leq& BT + V_r\sum_{t=rT}^{rT+T-1}g(t) + \sum_{t=rT}^{rT+T-1}(t-rT)q^{diff}\cdot|y(t)-z(t)|\\ &+ q(rT)\sum_{t=rT}^{rT+T-1}[y(t)-z(t)]\\ \leq& T\cdot C(T) + V_r\sum_{t=rT}^{rT+T-1}g(t) + q(rT)\sum_{t=rT}^{rT+T-1}[y(t)-z(t)],\end{aligned} \tag{29}$$

where $C(T) = B + D(T-1)$, $q^{diff} = \max_{t=0,1,\cdots,J-1}\{y(t), z(t)\}$ and D is a finite constant satisfying $D \geq \frac{1}{2}q^{diff}\cdot\max\{y(t), r(t)\}$. Note that the right hand side of (28), which is explicitly minimized by COCA, is smaller than or equal to that of (29). Thus, by applying COCA on the left-hand side and considering the optimal T-step lookahead policy on the right-hand side of (29), we obtain the following inequality

$$\triangle_T(rT) + V_r\sum_{t=rT}^{rT+T-1}g^*(t) \leq T\cdot C(T) + V_rTG_r^*, \tag{30}$$

where $g^*(t)$ is the cost achieved by COCA at time t. Note that $q(rT)$ is reset to zero, for $r = 0, 1, \cdots, R-1$, as enforced by Algorithm 1, whereas $\triangle_T(rT) = q^2(rT+T) - q^2(rT) = q^2(rT+T)$ in (30) is the T-step Lyapunov drift calculated *after* the r-th reset but *before* the $(r+1)$-th reset of the carbon deficit queue. Thus, before the $(r+1)$-th reset of the carbon deficit queue, we obtain from (30)

$$\begin{aligned}q(rT+T) &\leq \sqrt{BT + DT(T-1) + V_rT\left(G_r^* - g_{\min}\right)}\\ &= \sqrt{T}\cdot\sqrt{B + D(T-1) + V_r\left(G_r^* - g_{\min}\right)},\end{aligned} \tag{31}$$

where $g_{\min}$ is the minimum cost that can be achieved by any feasible decisions throughout the budgeting period. Then, by the inequality (27), we derive

$$\frac{1}{T}\sum_{t=rT}^{(r+1)T-1}y(t) \leq \frac{1}{T}\sum_{t=rT}^{(r+1)T-1}z(t) + \frac{\sqrt{C(T) + V_r\left(G_r^* - g_{\min}\right)}}{\sqrt{T}}, \tag{32}$$

where we define $C(T) = B + D(T-1)$. Therefore, by summing (32) over $r = 0, \cdots, R-1$ and dividing both sides by R, we prove part (a) of Theorem 2.

Next, by dividing both sides of (30) by V_r and considering $q(rT) = 0$ as enforced by Algorithm 1, it follows that

$$\begin{aligned}\sum_{t=rT}^{rT+T-1}g^*(t) &\leq \frac{BT}{V_r} + TG_r^* + \frac{DT(T-1)}{V_r} - \frac{\triangle_T(rT)}{V_r}\\ &\leq \frac{BT + DT(T-1)}{V_r} + TG_r^*.\end{aligned} \tag{33}$$

Thus, by summing (33) over $r = 0, \cdots, R-1$ and dividing both sides by RT, we prove part (b) of Theorem 2. ■