Electricity Bill Capping for Cloud-Scale Data Centers that Impact the Power Markets

Yanwei Zhang†, Yefu Wang†, and Xiaorui Wang†,‡
†Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN 37996
‡Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210
†{yzhang82, ywang38}@utk.edu ‡xwang@ece.osu.edu

***Abstract*—Minimizing the energy consumption of data centers has been researched extensively. However, much less attention is given to a related but different research topic: minimizing the electricity bill of a network of data centers by leveraging different electricity prices in different geographical locations to distribute workloads among those locations. Initial solutions to this problem are oversimplified with an unrealistic assumption that the huge power demands of data centers have no impact on electricity prices. As a result, they cannot be applied to cloud-scale Internet data centers that are expected to grow rapidly in the near future and can draw tens to hundreds of megawatts of power at peak. In addition, existing solutions focus only on server power consumption without considering cooling systems and networking devices, which account for up to 50% of the power consumption of a data center.**

In this paper, we propose a novel *electricity bill capping* algorithm that not only minimizes the electricity cost, but also enforces a cost budget on the monthly bill for cloud-scale data centers. Our solution first explicitly models the impacts of the power demands induced by cloud-scale data centers on electricity prices and the power consumption of cooling and networking in the minimization of electricity bill. In the second step, if the electricity cost exceeds a desired monthly budget due to unexpectedly high workloads, our solution guarantees the quality of service for premium customers and trades off the request throughput of ordinary customers. We formulate electricity bill capping as two related constrained optimization problems and propose efficient algorithms based on mixed integer programming. Extensive results show that our solution outperforms the state-of-the-art solutions by having lower electricity bills and achieves desired bill capping with maximized request throughput.

I. Introduction

High energy consumption has become one of the most serious concerns for large-scale data centers that are rapidly expanding the number of hosted servers. Therefore, minimizing the energy consumption of data centers has recently attracted a lot of research efforts. However, few studies have tried to address a related but different research topic: minimizing the electricity bill of a network of data centers by leveraging different electricity prices in different geographical locations to distribute workloads among those locations. This research topic is important for many Internet service providers, such as Google, Microsoft, and Yahoo!, to minimize their operating costs, because they commonly have massive and geographically distributed data centers to support various services such as cloud computing.

A few initial studies have been recently conducted to address the problem of electricity cost minimization [1], [2]. The key idea of those studies is to periodically monitor the time-varying electricity prices of the regions where data center sites are located. Based on the price information, Internet requests are routed to those sites where electricity prices are relatively low for minimized operating costs. While those studies have shown promise, they have two major limitations that prevent their applications to cloud-scale data centers that are expected to grow rapidly in the near future.

First, existing solutions rely on re-routing requests and turning on/off servers to control the power consumption of each data center site and thus the total electricity cost. However, they model only the power consumption of computer servers in their analyses, while increased workload and more active servers in a data center also lead to increased power consumption to run the cooling systems and networking devices [3], [4]. Recent studies show that cooling can take up to 25% [1] and network can account for up to 20% [4] of the total power consumption in a data center. The cooling and networking power consumption also varies significantly with the data center workload, especially in future energy-proportional data centers [5]. Therefore, those portions of power consumption must be considered in the cost minimization problem for correct and intelligent decision making.

More importantly, the second limitation is their unrealistic assumption that the huge power demands of data centers have no impact on electricity prices. In other words, data centers are treated simply as *price takers* in power markets and their electricity prices are assumed to be irrelevant to their power demands at a given time point. However, the reality in power market operation is that electricity prices are frequently adjusted mainly based on a well-known policy called the Locational Marginal Pricing (LMP) methodology [6]. According to LMP, electricity prices depend not only on geographical region and time, but also on the locational supply and demand of power. Therefore, while traditional small-scale enterprise data centers may be assumed to be passive price takers, this assumption is no longer valid for cloud-scale data centers whose sizes are much larger. For example, some data centers host more than 300,000 servers [7], [8] and can draw tens to hundreds of megawatts of power at peak. As a result, cloud-scale data centers become the major power consumers of power suppliers and thus are now *price makers*. To deal with the high power demands from those price makers, many power suppliers offer *Peak Power Rebate* pricing policies such that large power consumers get a temporarily lowered price for voluntarily reducing electricity use during peak times. For example, participants in the *Power Smart Pricing* program of

This work was supported, in part, by NSF under CNS-0720663 and CAREER Award CNS-0845390.

the Ameren Illinois Utilities could save an average of 20% with the locational pricing policy [9]. In addition, due to the transmission limitations of the power grid, some suppliers impose a cap on the power draw at different time scales (daily or monthly), and penalize those price makers heavily if this cap is exceeded. Thus, the power demands of cloud-scale data centers have significant impacts on electricity prices and the impacts must be addressed for minimized electricity bills.

Capping the electricity bill of could-scale data centers is another equally important issue for cloud-service providers. Since the electricity cost of operating data centers has become a significant portion (20% or more) of the monthly costs of those providers [8], it is a common business procedure for them to allocate a monthly budget for electricity cost. However, due to the high variations in data center workloads, it is usually difficult to enforce such a desired budget on electricity cost. For example, breaking news on major newspaper websites may incur a huge number of accesses in a short time and thus lead to unexpectedly high electricity costs for data centers. Note that cost minimization alone cannot enforce a desired electricity bill cap, because a monthly budget for electricity is commonly made based on history data with a certain safety margin. Therefore, if similar events occur frequently in a month and no effective methods are taken to control the cost, the monthly budget is likely to be violated.

To enforce a desired electricity bill cap in the face of unexpectedly high workloads, a service provider may need to differentiate premium customers who pay for their services from ordinary customers who enjoy complimentary services. The optimization objective is to guarantee the quality of service (*e.g.*, response time) for premium customers, while reducing (to the minimum degree) the request throughput of ordinary customers for lowered electricity use and costs. We argue that electricity bill capping is becoming an increasingly important issue, as cloud-scale data centers are rapidly expanding their sizes. Bill capping should be addressed together with power capping, which is recently proposed to cap the power consumption of a single data center [10], [11]. To cap the electricity use and bill of cloud-scale data centers, the power cap of each data center site must first be enforced to avoid financial penalty [12]. The total electricity cost of the entire data center network should then be controlled to avoid resulting in a high budget deficit. Electricity bill capping offers cloud-service providers a flexible and effective way to achieve maximized return within their sometimes stringent budget.

In this paper, we propose a novel electricity bill capping algorithm that conducts dynamic request dispatching to not only minimize the electricity bill, but also enforce a cost budget on the monthly bill for cloud-scale data centers. In the first step, our solution explicitly models the impacts of the power demands of cloud-scale data centers on electricity prices and the power consumption of cooling and networking in the minimization of electricity cost. In the second step, if the minimized electricity cost still exceeds the desired monthly budget due to unexpectedly high workloads, our solution guarantees the quality of service for premium customers and trades off the request throughput of ordinary customers. To our best knowledge, our work presents the first study on electricity bill capping for cloud-scale data centers. Specifically, the contributions of this work are three-fold:

- We propose to address a new and important problem, electricity bill capping, for cloud-scale data centers. While existing work only minimizes the cost in a best-effort manner, our algorithm explicitly enforces a cost budget and maximizes the request throughput of the distributed data centers within the budget.
- We consider realistic pricing policies in power markets and model the impacts of the power demands of cloud-scale data centers on electricity prices. We also take into account the power consumption of cooling and networking to minimize the electricity cost. As a result, our solution leads to lower costs than existing solutions on electricity cost minimization.
- We formulate electricity bill capping as two related constrained optimization problems and propose an efficient algorithm based on Mixed Integer Programming. Extensive results show that our solution outperforms the state-of-the-art solutions and achieves desired bill capping with maximized request throughput.

The rest of the paper is organized as follows. Section II provides background information on power pricing. Section III introduces the overall architecture of the proposed cost capping solution. Sections IV and V present the modeling and formulations of the two steps of cost capping, *i.e.*, cost minimization and throughput maximization, respectively. Section VI discusses the simulation strategy. Section VII introduces our extensive experimental results. Section VIII reviews the related work. Section IX discusses some challenges for future work and Section X concludes the paper.

II. Background on Power Pricing

In the power market, generators and consumers of power are usually connected to an electricity grid, a complex network of transmission and distribution lines. For example, the United States is divided into eight such grids. Furthermore, the electricity prices usually change as a function of the regional load variation due to the complex transmission conditions and different generator profiles in the grids [13]. In other words, electricity prices in the local power markets may exhibit fluctuations with the variable power demands, *e.g.*, a step change may show up in LMP when load grows to a certain level. The load increase may be caused by either a transmission line reaching its limit or a generator reaching its limit. For example, in the Pennsylvania-New Jersey-Maryland Interconnection (PJM) five-bus sample system [13], a step change of the prices happens with a system load of 600MW since the generator in the area of Brighton reached its output limits. Similarly, there is another LMP step change due to a new transmission limit of the line between areas of Brighton and Sundance at the system load of 711.81 MW.

In order to determine the electricity prices, LMP methodology has been used as a dominant approach in energy market operation and planning [6], [13]. A number of Independent System Operators (ISO), such as PJM, ISO-New England, have implemented or taken into considerations the LMP methodology to determine how the electricity prices change

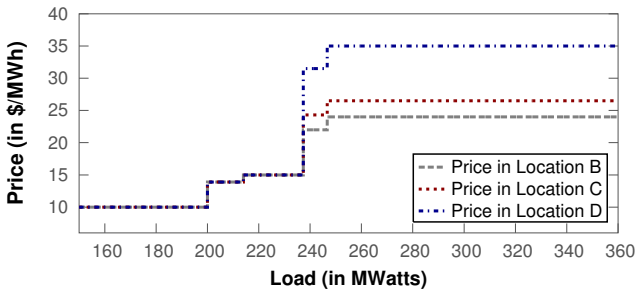

Fig. 1. Locational electricity pricing policies in three locations in the state of New York.

with the power demands in the local power markets [14]. In fact, the present LMP methodology leads to a step change when a new constraint, either transmission or generation, becomes binding as load increases. Figure 1 shows the locational pricing policies in the three locations of B, C, and D from the well-known PJM five-bus system [13], with respect to all the loads and generation supplies connected to the PJM system. This figure is derived from the study on the LMP methodology utilized in the real-world power markets [6]. Specifically, the five-bus PJM system is composed of five generators and three distributed consumers, referred to as B, C, and D. The five generators are located in the areas of Alta, Park City, Solitude, Sundance, and Brighton in the state of New York, respectively. Furthermore, the system load is uniformly distributed at the three distributed consumers. Thus, a specific locational pricing policy can be derived from Figure 1 for each local power market of the three distributed consumers.

On the other hand, as discussed before, due to continuously increasing service demands from customers, cloud-service data centers are rapidly expanding their sizes. Some have already approached the order of hundreds of thousands or more servers that can draw tens of megawatts of power at peak [7], [8]. As a result, cloud-scale data centers become major power consumers of power suppliers and thus are now price makers in the power markets. This reality is in sharp contrast to the unrealistic assumption in the existing research [1], [2] that the huge power demands of data centers have no impact on electricity prices. The unawareness of cloud-scale data centers playing the role of *price maker* may result in sub-optimal cost minimization efforts, as demonstrated in Section IV.

III. System Architecture

In this section, we provide a high-level description of our bill capping solution that adaptively allocates workloads among geographically distributed data centers using the locational pricing policies.

In our work, we assume that a network of data centers share a cost budget in every budgeting period determined by the administration departments of the owner of the Internet applications. We also assume that the locational pricing policies, *i.e.*, how the changes in power consumption of data centers affect the electricity prices in local power markets, are available from the ISO. For example, the electricity price may be derived as a function of the total power consumed by all customers in the same ISO region, based on the specific algorithm that the ISO is using [13]. As shown in Figure 2, the key components in our cost management framework include a centralized *bill capper* and *budgeter* that are invoked periodically in every *invocation period*. In this paper, we use one month as the budgeting period and one hour as the invocation period. Those invocation periods are suggested to be good trade-offs between management granularity and actuation overheads [3] for data center-level management algorithms.

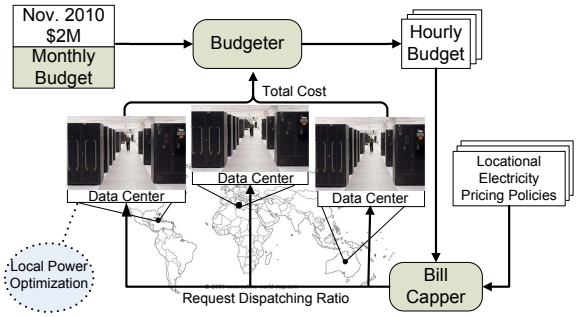

Fig. 2. Proposed electricity cost capping architecture for distributed cloud-scale data centers.

When the budgeter receives a monthly budget at the beginning of the budgeting period from the system administrator, it breaks the monthly budget into hourly budgets based on the historical incoming workload data. In particular, at the beginning of every invocation period, the hourly budget is computed based on the monthly cost budget from the service provider and the electricity cost already consumed in the previous invocation periods, as well as the observations of the workload's historical behaviors in the same hours in the past (*e.g.*, last two weeks) as discussed in Section VI-B. Then, the bill capper determines the workload allocations such that:

- The total electricity cost of data centers is minimized and is below the budget of the current hour determined by the budgeter.
- The application-level quality of service (QoS) of customers is provided in a best-effort manner. That is, if the budget allows, all customers achieve their application-level performance targets. If the budget is too low, the QoS of premium customers is guaranteed while the QoS of ordinary customers is provided in a best effort manner within the cost budget.

As discussed above, our cost capping algorithm includes two steps. (1) In the first step, the algorithm solves a ***cost minimization*** problem that minimizes the total cost of data centers with the consideration of the locational pricing policies, by distributing the Internet requests to different data centers in an efficient way; (2) In the second step, the algorithm compares the computed cost found in the first step with the given hourly budget. If the computed cost is below the budget, the workload allocations determined in the first step is enforced. Otherwise, the algorithm solves a ***throughput maximization within cost budget*** problem that determines an admission rate to enforce admission control only for requests from ordinary customers. The capping algorithm also determines the web request allocation to every data center such that the total cost of data centers is controlled below the cost budget. Once the workload allocations (*e.g.*, the fraction of workload allocated to each data center) are determined by the bill capper, (*i.e.*, either in Step (1) or Step (2), depending on the allocated cost budget), the dynamic request routing mechanism in the

cloud-scale data center networks dispatches the incoming requests among data centers based on the determined request dispatching strategy. Note that dynamic request routing and mechanisms to replicate the data at multiple data centers have already been implemented in the cloud-scale data center networks by many Internet service providers to map requests to servers, for the purposes of customer QoS guarantees and fault-tolerance [1]. For example, the Authoritative Domain Name System (DNS) is deployed to take the request dispatcher role by mapping the request URL hostname into the IP address of the destined data centers [15]. It is important to note that the adopted request dispatching strategy does *not* migrate or redistribute any request among different data centers once the request is dispatched to a data center. In addition, we assume that each data center has a local optimizer to dynamically minimize the number of active servers in the data center based on the performance model discussed in Section IV-B, given the distributed workload.

We introduce the two steps of the proposed bill capping algorithm in detail, ***cost minimization*** and ***throughput maximization within cost budget***, in the following sections.

IV. Cost Minimization

In this section, we present the system modeling and problem formulation of the first step, cost minimization.

A. Problem Formulation

We first introduce the following notation. A cloud-scale data-center system consists of N data centers . The i^{th} data center is located in the i^{th} location and consumes p_i watts of power in an invocation period. The power consumption of the i^{th} data center should not exceed a power constraint of Ps_i. Pr_i is the electricity price in the power market at the i^{th} location. The electricity power price is a known function of the total power consumption of P_i in the same ISO, *i.e.*, $Pr_i = F_i(P_i)$. The power consumed by all consumers other than the data center is d_i. Consequently we have $P_i = p_i + d_i$. The whole data center system has a workload of λ requests per hour. Our algorithm allocates the i^{th} data center with λ_i requests per hour. The average response (or delay) time of the i^{th} data center is R_i and Rs_i is the corresponding performance set point.

Given a workload of λ requests per hour, the goal of the cost minimization problem is to dynamically choose a request allocation strategy such that the i^{th} data center is assigned with λ_i requests per hour ($0 \leq i \leq N$) to minimize the overall electricity cost of the N data centers

$$Minimize: \sum_{i=1}^{N} Pr_i \cdot p_i \tag{1}$$

$$\text{such that } (a)\sum_{i=1}^{N} \lambda_i = \lambda; (b) p_i \leq Ps_i; (c) R_i \leq Rs_i \tag{2}$$

Specifically, p_i (in MW) will be numerically the same as energy (in MWh) since the invocation period used in our cost capping algorithm is assumed to be one hour.

In order to solve the optimization problem in (1 - 2), it is important to model the variables Pr_i and p_i as a function of the request distribution λ_i. Since we have $Pr_i = F_i(p_i + d_i)$ based on the pricing policies and assume that d_i is periodically informed by ISO, we model the power consumption p_i and the average performance R_i for the i^{th} data center as follows. Please note that the key contribution of our paper is the optimization framework and methodology for electricity bill capping for cloud-scale data centers. To this end, due to the limited space and our focus on the optimization framework and methodology, we adopt some simplified but well-established power models in this work. In particular, the models used in this work have been commonly verified in some recent studies such as [3], [16], [17], [18], [4], [19]. Furthermore, without loss of generality, our framework can be easily integrated with more detailed and sophisticated power models.

B. Performance and Power Models

Queueing theory is commonly used in modeling system performance, such as in [3], [16], [17]. In this paper, we model a data center as a G/G/m queue [3] to account for different response time among different data centers in workload dispatching. That is, a single data center is considered as an m-server queueing system, where each server has a generalized service time distribution to provide service for incoming requests with a generalized arrival and request-size distribution. In particular, according to the well-known Allen-Cunneen approximation [20], [21] in modeling the G/G/m queue, we have

$$R_i = \frac{1}{\mu_i} + \frac{C_A^2 + C_B^2}{2} \cdot \frac{(\rho^{n_i} + \rho)/2}{n_i \cdot \mu_i - \lambda_i} \tag{3}$$

where $\rho = \lambda / n_i \mu$ describes the average utilization of a server in the data center. C_A^2 and C_B^2 represent the squared coefficient of variation of request inter-arrival time and request sizes, respectively. Specifically, the average request arrival rate and request sizes can be monitored by the bill capper in order to characterize these two factors, *i.e.*, C_A^2 and C_B^2.

As shown in equation (3), the average response time for the requests serviced in the data center consists of two portions: 1) the service time, *i.e.*, $\frac{1}{\mu}$, given the service rate μ of a single server in the data center and 2) the average waiting time that the requests spend in a queue waiting to be serviced. Due to the fact that the number of the servers calculated in equation (3) is the minimal number required to provide guaranteed service for the incoming requests to the data center, all the active servers in the data center will likely keep busy. Therefore, we have ρ approximates 1. Again, this approximation is based on the fact that in this work a local optimizer is assumed to be running in each data center in order to minimize the number of active servers. Hence, the average waiting time for a request, *i.e.*, the second term in equation (3) can be replaced as $(\frac{C_A^2 + C_B^2}{2})(\frac{1}{n_i \cdot \mu_i - \lambda_i})$, which is also adopted by a recent study to model the response time and the number of servers needed to satisfy a given demand [2].

We model the power consumption of a data center as the sum of three portions, power consumed by servers, cooling systems and networking devices, since those three portions account for 80% - 90% dynamic power consumption of cloud-scale data centers [3].

$$p_i = p_i^{server} + p_i^{networking} + p_i^{cooling} \tag{4}$$

Power model for servers. As indicated in Figure 2, every data center runs a local optimizer that dynamically adjusts the number of active servers to provide a desired level of QoS (*i.e.*, response time) with the least number of servers. As a result, given a request rate λ_i and a desired response time Rs_i, the number of desired active servers n_i can be derived from (3). The power consumed by all the active servers in the data center is then modeled as

$$p_i^{server} = \sum_{k=1}^{n_i} sp_i^k \tag{5}$$

where sp_i^k is the power consumption of the k^{th} active server in the i^{th} data center. In particular, the power consumption of a single server is usually a linear function of server utilization [3]. That is, $sp = I + D \cdot u$, where I denotes the server idle power, D denotes the server power at 100% utilization, and u denotes the utilization level. In order to model the single server power in this work, sp is calculated with respect to the actual server utilization level (*e.g.*, 80%) by the local optimizer in each data center. Note that equation (5) is a general total server power consumption formula useful for both heterogeneous and homogeneous data centers. In particular, in order to capture the power consumed by all the active servers in a homogeneous data center, equation (5) can be formulated as $p_i^{server} = n_i \cdot \bar{sp}_i$, where $\bar{sp}_i$ is the averaged power consumption of a single server in the i^{th} data center.

Power model for networking devices. Typical architectures in today's data center network topologies are composed of three-level trees of switches or routers [18]. Specifically, it has a core level as the root of the network topology tree, an aggregation level in the middle and an edge level as the leaves [4]. In our work, we adopt a commonly used three-level topology called a *k-ary fat-tree* to connect Ethernet switches in data centers as in a recent study [18]. Accordingly, the power consumption by networking devices is estimated as

$$p_i^{networking} = A_i \cdot esp_i + B_i \cdot asp_i + C_i \cdot csp_i \tag{6}$$

where esp_i, asp_i, and csp_i are the average power consumption of an edge switch, an aggregate switch, and a core switch, respectively. A_i, B_i and C_i are proportional to the number of the active servers based on the value k of the fat-tree topology network. We further assume that esp_i, asp_i, and csp_i are constant since today's network elements are not energy proportional, *e.g.*, a switch going from zero to full traffic increases power by less than 8% [4]. Thus, the power consumption of networking devices is a function of the number of active switches, which vary significantly based on data center workloads [4].

Power model for cooling systems. The power consumed by cooling systems in a data center depends on the cooling strategies used, the weather conditions, and the power consumed by the IT equipment. We assume that an efficient cooling strategy related to the external air [19] runs in the data center and provides a certain value of cooling efficiency coe_i, defined as the heat being removed by the cooling systems used in the data center relative to the power consumed by the systems. A lower temperature of the external air around the data center means a higher value of coe_i and more efficient cooling. The cooling power consumption is then estimated based on the model proposed by Ahmad et al. [3].

$$p_i^{cooling} = coe_i^{-1} \cdot (p_i^{server} + p_i^{networking}) \tag{7}$$

C. Solution Design

Based on the analysis above, cloud-scale cost minimization has been modeled as a constrained optimization problem. In particular, the optimization problem formulated in (1) - (7) is non-linear since the pricing policies involved in $Pr_i = F(p_i + d_i)$ are usually non-linear [13], [6]. As discussed in Section II, pricing policies in local power markets, referred to as Pr_i, are typically a piece-wise function of the total power consumption P_i in the same ISO.

Therefore, in order to solve the optimization problems, we leverage a standard technique discussed in [22] to formulate the problems in (1) - (9) as Mixed Integer Linear Programming (MILP) problems, since the only non-linear part in our optimization problem is a piece-wise function of the locational pricing policies Pr_i. Specifically, we introduce $m_i - 1$ logic and $m_i - 1$ real variables for each Pr_i, where m_i represents the number of different price levels in the i^{th} location. The logic variables are used to define which price level is chosen with respect to different loads while the real variables are to define the corresponding electricity prices. The transformation is not shown due to space limitations, but the details can be found in [22]. After the transformation, a standard MILP solver (*e.g.*, lp_solver) [23], which is widely used for optimization of various problems in industry, is used on-line to solve the optimization problems in this and the next sections. Specifically, lp_solver uses a branch-and-bound algorithm to solve MILP problems. The computational complexity of lp_solver is exponential to the number of the binary variables, *i.e.*, the total number of price levels in all the pricing policies. Fortunately, cloud-scale large systems typically own only a limited number of data centers [24]. For example, Facebook operates just about 10 data centers. Furthermore, there are just several (*e.g.*, 5) different pricing levels in the real-world pricing policies [6]. Based on the simulation in Section VI, for a large system with 3 data centers and 5 different pricing levels, lp_solver consumes at most 21 millisecond in an invocation period of one hour to determine the optimal workload allocations with up to 10^8 requests.

V. Throughput Maximization within Budget

In our work, the proposed cost capping algorithm guarantees the QoS (*i.e.*, response time) for premium customers and trades off the request throughput of ordinary customers if the monthly budget is likely to be exceeded. Specifically, the second step in the cost capping algorithm is to determine an admission rate to enforce admission control only for Internet requests from ordinary customers. Our algorithm then determines the requests distributed to each data center such that the total cost is controlled to stay below the given cost budget. The key rationale is that premium customers are the revenue source for cloud service providers since they pay for the required service. In order to maintain the primary financial source for the business, the service providers have to guarantee the QoS

for premium customers; otherwise, they may lose the revenue source due to the unsatisfactory service.

In addition to the notation already introduced in Section IV, we define more here. Cs denotes the desired cost constraint in an invocation period determined by the budgeter and C_i is the electricity cost of the i^{th} data center , *i.e.,* $C_i = Pr_i \cdot p_i$. Given an hourly cost budget of Cs from the budgeter, the goal of the throughput maximization problem is to dynamically determine the request allocations such that the i^{th} data center is assigned with λ_i requests per hour ($0 \leq i \leq N$) to guarantee the QoS for premium customers and the maximized request throughput for ordinary customers within the cost budget. We now formulate the problem as follows:

$$Maximize: \sum_{i=1}^{N} \lambda_i \tag{8}$$

$$\text{such that} (a) \sum_{i=1}^{N} C_i \leq Cs; (b) p_i \leq Ps_i; (c) R_i \leq Rs_i \tag{9}$$

It is important to note that the cost capping algorithm guarantees the QoS for premium customers despite an insufficient cost budget and a best-effort throughput will be provided to ordinary customers with the remaining cost budget after servicing premium requests. This corresponds to two situations: 1) Cost budget Cs is sufficient to guarantee the QoS for all premium customers and can still service some ordinary requests, and 2) Cs is too stringent to even provide QoS guarantee for premium customers. In the second situation, the budget has to be violated because the QoS of premium customers must be guaranteed. In the next two subsections, we discuss the strategies used in the two situations, respectively.

A. Sufficient Cost Budget

The objective of the optimization problem in (8) is to choose a request allocation scheme, such that the i^{th} data center is assigned with λ_i requests per hour ($0 \leq i \leq N$) to maximize the overall throughput of the N data centers within the given cost budget Cs. It is clear that the total assigned requests to the N data centers should not exceed the arrival workload of λ. If the overall throughput $\lambda^{throughput}$ is not lower than the premium web requests, referred to as $\lambda^{premium}$, the workload allocations determined are enforced as the solution to the optimization problem in (8). That is,

$$\lambda^{throughput} = \sum_{i=1}^{N} \lambda_i$$

In this case, all the premium requests are guaranteed QoS, and a maximal throughput of $\lambda^{ordinary}$ is provided to ordinary customers.

$$\lambda^{ordinary} = \lambda^{throughput} - \lambda^{premium}$$

B. Insufficient Cost Budget

Despite an insufficient cost budget for premium customers due to reasons like unexpectedly high workloads, *i.e.*, $\lambda^{throughput} < \lambda^{premium}$, service providers have to guarantee the QoS for all the premium customers. Accordingly, the optimization problem in (8) will be reconfigured as a cost minimization problem in the form of (1 - 2) with the workload of $\lambda^{premium}$, instead of λ.

In this case, no services are provided to ordinary customers. In fact, the given cost budget Cs is exceeded in such invocation periods due to the QoS guarantee for premium customers.

VI. Simulation Strategy

Given the limitations on hardware facilities, we could not construct a data center that has power consumption high enough to change electricity prices. However, we employ real-world data center traces and realistic server configurations to evaluate our technique. Note that the similar evaluation methodology has been commonly used, such as in [2], [3]. In particular, we use real-world web request traces, as well as a power consumption trace from the real-world power market to simulate a locational power consumption by power consumers other than data centers, to evaluate the performance of the proposed cost capping algorithm. These evaluation primarily target web server-based applications, which have been widely adopted for evaluations in data cener-related simulations.

A. Datacenter Parameters

In our evaluation, we simulate a cloud-scale system composed of three geographically distributed data centers for a cloud service provider. Each data center hosts up to 300,000 servers, which is consistent to the disclosed scale of the data centers, e.g., operated by Microsoft [7]. We assume that the power consumption profile for each server at the same location remains constant, which is usually true when homogeneous servers and configurations are used in each data center [2]. Specifically, the server configuration in each location is respectively assumed to be as follows [19]: Data Center 1 (2.0 GHz AMD Athlon processor), Data Center 2 (1.2 GHz Intel Pentium 4630 processor), and Data Center 3 (2.9 GHz Intel Pentium D950 processor). Their power consumption is assumed to be 88.88, 34.10, and 149.19 Watts and their processing capacity coefficients are estimated as 500, 300, and 725 requests per second, respectively. The average edge switch power, aggregate switch power, and core switch power are assumed to be (184, 184, 240), (170, 170, 260), and (175, 175, 240) Watts for the three simulated data centers [4].

B. Real-World Traces

To build our workloads in the simulator, we use a trace of Internet traffic from Wikipedia.org [25]. In particular, we use this tracefile with a 2-month long data, which contains 10% of user requests arrived at Wikipedia between October 1st, 2007 and November 30th, 2007. Since the numbers of requests in the original trace file are 10% of user requests arrived at Wikipedia, we proportionally increase the request numbers by multiplying with a scaling factor (*i.e.*, 10) in our simulation to emulate the accurate number of the incoming requests. Specifically, the users' behavior in the trace shows a very clear weekly pattern in visiting the Wikipedia website. Thus, we take the 1-month long Wikipedia trace of November as the incoming workload in the simulator while using the October trace data to work as the historical observations of the workload to predict hourly cost budgets. To do so, we maintain

a history of the request arrival rate seen during each hour of the week over the past several weeks. We then calculate every averaged hourly workload weight of the whole week over the past several weeks as the hourly budget weight in the coming week. Based on experiments, we find that for this Wikipedia trace, a 2-week long history trace data can provide a reasonable prediction on hourly cost budgets. Note that more sophisticated prediction methods, such as [26], can also be integrated into our system.

The simulator also uses a power consumption trace file from the real-world power market to simulate a locational power consumption by power consumers other than data centers. The data is collected in the location of Rockland Electric (RECO) in PJM system, from June 1 through June 30, 2005 [27]. Additionally, in order to simulate the cooling strategy run in data centers discussed in Section IV-B, we refer to the cooling efficiencies as, 1.94, 1.39, and 1.74 for the three data centers [19], respectively. The pricing policies used in our simulation are generated based on the well-known PJM five-bus system as Figure 1.

VII. Evaluation Results

In this section, we first introduce a state-of-the-art baseline. We then compare our electricity bill capping algorithm (referred to as Cost Capping) against the baseline.

A. Baseline and Electricity Price

In our work, we use a state-of-the-art cost minimization algorithm, referred to as *Min-Only*, as the baseline in our experiments. Min-Only is an optimization-based cost minimization algorithm designed for Internet-scale data centers [2]. Min-Only represents a typical research solution to minimize the electricity bill for the service providers who operate large-scale data centers.

There are three fundamental differences observed between Cost Capping and the Min-only strategy. First, Min-only has an unrealistic assumption that the resulting workload allocations to data centers will *not* have impact on the locational electricity price in the local power market of each data center; Second, Min-Only focuses only on server power consumption without considering cooling systems and networking devices. Finally and most importantly, since Min-Only only tries to minimize the electricity cost without throttling the request throughput of ordinary customers, it may exceed the desired cost budget in the face of heavy requests.

The Min-Only strategy assumes a constant locational electricity price for each data center in an invocation period. However, the real electricity price is actually a function of power demand, as show in Figure 1. Thus, to compare with Min-Only, we adopt two different methods to simulate electricity prices for Min-Only with respect to the real-world locational pricing policies used in our work, referred to as Min-Only (Avg) and Min-Only (Low), respectively. For Min-Only (Avg), the price strategy is assumed as the averaged value of all the step prices; for Min-Only (Low), the price strategy is simulated as the lowest step price. For example, the electricity price ($\$/MWh$) of Min-Only (Avg) in Data Center 1 equals $16.98 = (10.00 + 13.90 + 15.00 + 22.00 + 24.00)/5$, while it is 10.00 for Min-Only (Low), based on the locational pricing policy in Figure 1.

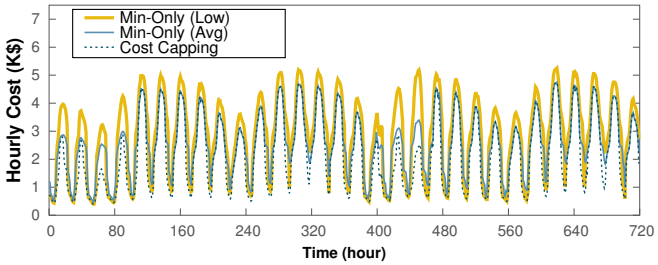

Fig. 3. Hourly electricity cost comparison between Cost Capping and Min-Only with respect to Nov. 2007 Wikipedia trace.

We use Min-Only (Avg) and Min-Only (Low) to show that a well-designed cost minimization algorithm with the assumption that the data centers are simply the price takers in power markets will lead to sub-optimal workload allocations and thus an unnecessarily high electricity bill. It will also violate the cost budgets easily in the face of heavy workloads from customers.

B. Electricity Cost Minimization

In this experiment, we compare the first step of Cost Capping with Min-Only in terms of minimized electricity cost.

Figure 3 shows the comparison of hourly electricity cost resulting from Cost Capping and Min-Only with the Wikipeida workload. As can be seen, the electricity cost by Cost Capping is greatly reduced hourly, compared to the baselines. Specifically, Cost Capping has a (17.9%, 33.5%) more cost savings than Min-Only (Avg) and Min-Only (Low), respectively. That is, an up-to-$524M monthly electricity cost saving can be achieved by Cost Capping. As discussed in Section II, the key reason for Cost Capping to have lower costs is that Cost Capping uses the locational pricing policies in the process of determining optimal workload allocations to data centers.

Figure 4 illustrates the cost saving results of running Cost Capping and Min-Only under a series of different pricing policies from Pricing Policy 0 to Pricing Policy 3. Specifically, Policy 0 represents the case where the workload routing behavior from data centers has no impact on local power markets, *i.e.*, the case assumed by Min-Only; Policy 1 is the basic locational pricing policies derived from the five-bus PJM system, as discussed in Section II; Policies 2 and 3 are designed to double and triple, respectively, the price increase of Policy 1 when the the load is higher than 200MW. For example, the electricity prices ($\$/MWh$) in Data Center 1 based on Policy 1 are (10.00, 13.90, 15.00, 22.00, 24.00) with respect to the power load, while the prices based on Policies 2 and 3 are (10.00, 17.80, 20.00, 34.00, 38.00) and (10.00, 21.70, 25.00, 46.00, 52.00), respectively. Each data bar in Figure 4 represents the total electricity bill in the month under different cost management strategies and pricing policies. As shown in this figure, Cost Capping and Min-Only can gain the same electricity cost savings with Pricing Policy 0, since the workload allocations will *not* affect the electricity prices in the local power markets under this policy. With all the other pricing policies, Cost Capping results in a lower electricity bill, compared to Min-Only, due to the fact that it considers the locational pricing policies in the real-world

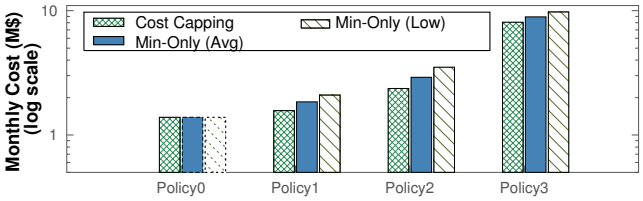

Fig. 4. Monthly electricity bills comparison under different pricing policies with respect to Nov. 2007 Wikipedia trace.

power markets. One may think that Cost Capping's gain of a lower electricity bill is at the expense of worse application performance. Our results show that Cost Capping achieves the same QoS guarantees as Min-Only, *i.e.*, the response time. The reason is that Cost Capping enforces a response-time performance constraint to guarantee the QoS for customers, as discussed in Section IV.

C. Throughput Maximization within Cost Budget

In this experiment, we test Cost Capping and Min-Only in a scenario that the cloud-service provider has a stringent cost budget to enforce. In this case, the second step of Cost Capping is invoked, to determine the workload allocations to data centers with guaranteed QoS for premium customers and a best-effort request throughput for ordinary customers, as discussed in Section V. We define *throughput* as the serviced requests with guaranteed QoS for customers.

In order to examine that Cost Capping can guarantee service to all premium requests while trying to enforce the limitation of the real-world cost budgets from data center administrators, we assume that 80% of the web requests from the trace file in each hour are generated by premium customers and 20% are from ordinary customers. Note that this specific proportion is orthogonal to our algorithm and other methods to define premium users can be easily integrated. We further assume a monthly cost budget of $\$2.5M$, which is estimated based on the workload in Nov. 2007 from the Wikipedia website.

Figures 5 and 6 illustrate how Cost Capping works under a monthly cost budget of $\$2.5M$. We can see that all the incoming requests from both premium and ordinary customers in each hour are guaranteed with service within the given cost budget, since the incoming request rate is relatively light with respect to the given monthly cost budget. This indicates an abundant cost budget. It is shown in two folds: 1) The throughput for both premium customers and ordinary customers is exactly the same as their input as in Figure 5; and 2) The electricity cost in each hour is below the given cost budget, as shown in Figure 6. In addition, Figure 6 shows that within one week the allocated hourly cost budget is in a growing way. This is due to the fact that we carry over the unused allocated cost budget from previous invocation periods to the remaining invocation periods in the same week.

As shown in Figures 7 and 8, with an insufficient monthly cost budget (*e.g.*, $\$1.5M$) to service the incoming requests from all the customers, all the premium requests still have guaranteed QoS, regardless of the given cost budget. These two figures also illustrate that Cost Capping provides a best-effort throughput for ordinary customers while controlling the electricity cost within the given cost budget. There are two interesting observations. First, for those invocation periods where no ordinary requests are serviced, *e.g.*, the hours of 176, 177, 178, 202 in Figure 7, they occur because no cost budget is left after servicing premium customers. For some of those invocation periods, the hourly electricity cost may exceed its hourly cost budget due to the mandatory QoS guarantees for premium customers, *e.g.*, the hours of 176, 177, as shown in Figure 8. Second, for those invocation periods where certain ordinary requests are serviced, *e.g.*, the hours of 13, 14, 15 in Figure 7, they occur because there is still some budget left after servicing all the premium requests. Therefore, a maximal throughput is provided to ordinary customers by Cost Capping with the electricity cost being controlled.

Figure 9 demonstrates the cost and throughput of Cost Capping and Min-Only with respect to a stringent monthly budget, *e.g.*, $\$1.5M$. Specifically, for the comparison on the monthly electricity bill, the results are normalized against the given monthly budget; and for the throughput comparison, the results are normalized against Min-Only (*i.e.*, all the incoming requests are serviced in Min-Only regardless of the given cost budget). Figure 9 shows that Min-Only can provide full service (*i.e.*, 100% throughput) for both premium customers and ordinary customers. However, due to the unawareness of the stringent cost budget, Min-Only (Avg) and Min-Only (Low) exceed the monthly cost budget by 23.3% and 39.5%, respectively. On the other hand, Cost Capping can guarantee a 100% throughput for premium customers and achieve an up-to-80.3% throughput for ordinary customers, while providing an accurate control on the electricity bill. That is, Cost Capping provides a 98.5% utilization on the given monthly cost budget.

We then study Cost Capping under a series of different monthly cost budgets. Figure 10 shows the monthly throughput under different monthly cost budgets. The results are normalized against the incoming premium requests and ordinary requests, respectively. It is clear that all the incoming requests from the premium customers are serviced with guaranteed QoS regardless of the given cost budget due to the QoS guarantee for premium customers in this work. Specifically, as shown in Figure 10, with an insufficient cost budget (*e.g.*, $\$500K$, $\$1.0M$ and $\$1.5M$), a best-effort throughput for ordinary customers is provided. For example, when the cost budget increases from $\$500K$ to $\$1.0M$ and $\$1.5M$, the throughput of ordinary customers increases from 94 million to 2.3 and 13 billion requests. When the cost budget is sufficient, *e.g.*, $\$2.5M$, all the incoming requests are serviced with guaranteed QoS. An interesting case is at the cost budget of $\$2.0M$, where some ordinary requests are not serviced despite the fact that the cost budget is not used up. The key reason is that in some invocation periods, the pre-allocated cost budgets are insufficient with respect to the incoming requests due to the workload's historical behavior-based budgeting strategy we used. As a result, some ordinary requests are not serviced in those periods. However, the number of un-serviced ordinary requests is just limited to 0.99% of the total incoming requests from the ordinary customers.

VIII. Related Work

Electricity cost is becoming a major proportion of cloud-scale data-center operating costs. Recently, a number of research studies have been conducted to optimize either the total

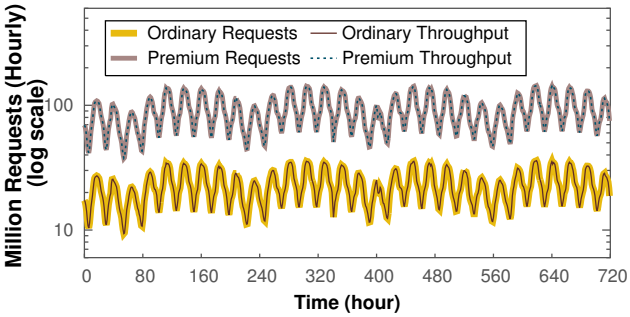

Fig. 5. Throughput by Cost Capping under a monthly cost budget of $\$2.5M$ with respect to Nov. 2007 Wikipedia trace.

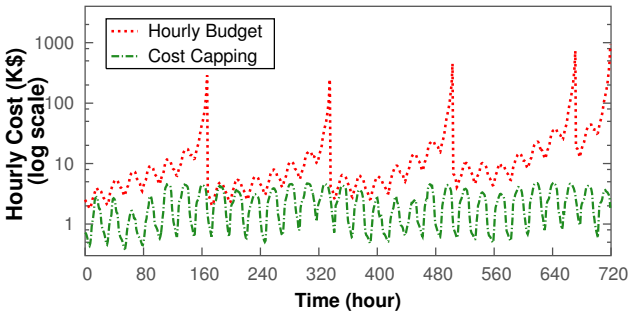

Fig. 6. Hourly electricity cost capping by Cost Capping under a monthly cost budget of $\$2.5M$ with respect to Nov. 2007 Wikipedia trace.

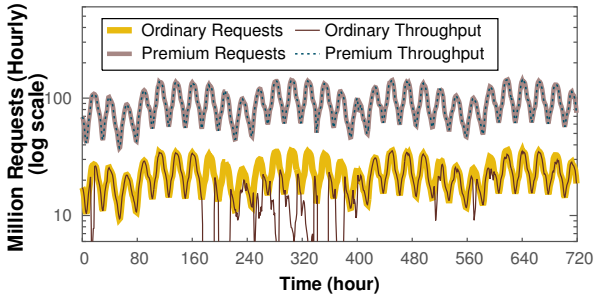

Fig. 7. Throughput by Cost Capping under a monthly cost budget of $\$1.5M$ with respect to Nov. 2007 Wikipedia trace.

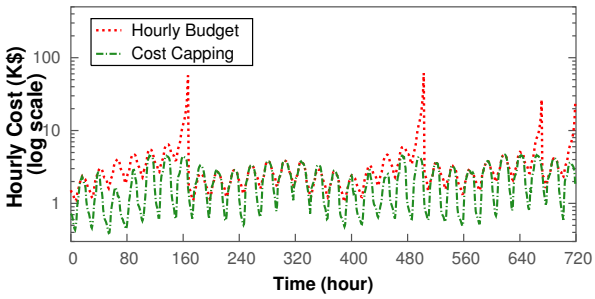

Fig. 8. Hourly electricity cost capping by Cost Capping under a monthly cost budget of $\$1.5M$ with respect to Nov. 2007 Wikipedia trace.

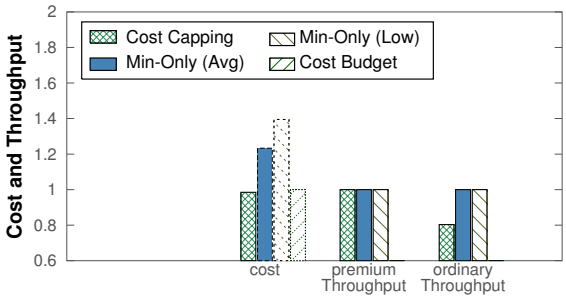

Fig. 9. Cost and throughput comparisons under a monthly cost budget of $\$1.5M$ with respect to Nov. 2007 Wikipedia trace.

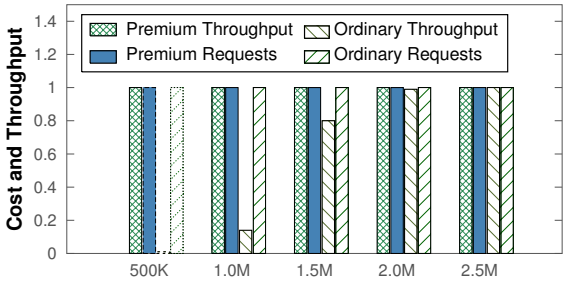

Fig. 10. Monthly throughput by Cost Capping with a series of monthly cost budgets with respect to Nov. 2007 Wikipedia trace.

energy consumption or the electricity cost of data centers. Many recent research projects have tried to minimize the energy consumption of data centers [28], [29], [30], [3], [4], [31]. However, all the aforementioned studies focus mainly on reducing the total energy consumption instead of the electricity bills.

A few recent projects have proposed to minimize the electricity bills of data center networks. For example, Qureshi et al. [1] try to lower the electricity bill by utilizing different electricity prices of the distributed data centers. Rao et al. [2] consider a multi-electricity-market environment to reduce the electricity bill. In a recent technical report [19], Li et al. propose to minimize the electricity bill of server facilities as well as cooling systems. Le et al. [32], [33], [34] propose to manage the brown energy consumption and leverage green energy for data center networks, while minimizing energy costs, in which they assume two electricity prices at each data center, one for "on-peak" hours and another for "off-peak" hours. In [35], Goiri et al. propose an optimization framework to automatically place datacenters for Internet service providers, by modeling response time, capital and operational costs, and cardon dioxide emissions. In another work [36], Le et al. investigate policies for virtual machine migration across data center networks, to lower electricity costs. In addition, Urgaonkar et al [37] and Govindan et al [38] explore the opportunities in reducing server power bill, by tapping into stored energy in data centers. In particular, a single data center is considered, instead of a data center network. However, none of these studies have considered the real-world pricing policies, *i.e.*, they make an unrealistic assumption that the request routing decisions of data center operators will *not* affect the locational electricity prices. More importantly, none of the existing research studies have tried to address the *electricity bill capping* issue.

IX. Discussion

There are several possible extensions to the proposed electricity bill capping framework. We briefly describe them here.

In this work, we stress our key contribution to conducting a novel study on data center electricity bill capping and the impacts of cloud-scale data centers on power prices. In order to verify our observations, we assume that homogeneous servers are used in a single data center, where the power and energy management for such a data center network could be simplified in determining the minimum number of active servers to provide service for the incoming requests. Unfortunately, due to the rapid development of high-performance CPU technologies, and data center repair, replacement, and

expansion, some data centers may not have such an ideal homogeneous configuration. For example, multiple service rates exist due to the heterogeneity in hardware. As a result, power and performance management is more complicated for a heterogeneous data center on how to distribute incoming request to different servers and how to dynamically configure the data center in determining the minimum number of active servers. Furthermore, the heterogeneity of the individual request in data-transfer requirements, as well as various data center applications, can be another non-trivial reality. We hope to address these challenges in our future work.

The proposed electricity bill capping architecture in this work is currently working in a centralized way to manage a data center network for minimizing the electricity cost as well as enforcing a cost budget on the monthly bill for cloud-scale data centers. While such a centralized architecture is commonly used in the management of data center networks [2], [1], it may not have a good scalability due to several reasons. First, the computational complexity of the bill capping algorithm depends on the number of data centers in the data center network as well as the number of different price levels in the pricing policy, and thus may not scale well for much larger-scale data center networks. Second, a centralized dispatcher may have long communication delays in larger-scale systems. Extending the electricity bill capping architecture to work in a hierarchical way is our future work. On the other hand, the proposed electricity bill capping scheme in this work is currently based on the assumption that there is an accurate enough prediction algorithm deployed in the system to forecast future incoming workload, which is consistent to the fact that there are some sophisticated algorithms that do workload prediction. However, in order to make our scheme more robust, in our future work we will improve our scheme to adapt to the situation when the workload prediction is inaccurate from time to time.

X. Conclusion

Existing work on electricity cost minimization oversimplifies the problem with an unrealistic assumption that the huge power demands of data centers have no impact on electricity prices. As a result, they cannot be applied to cloud-scale data centers that are expected to grow rapidly in the near future and can draw tens to hundreds of megawatts of power at peak. In this paper, we have presented a novel electricity bill capping algorithm that not only minimizes the electricity bill, but also enforces a cost budget on the monthly bill for cloud-scale data centers. Specifically, our solution achieves up to 33.5% more cost savings in the minimization of electricity bill by modeling the impacts of power demands on electricity prices. Furthermore, when the cost budget is too stringent to guarantee the QoS for all the customers, our bill capping solution can effectively control the electricity bill below the given cost budget, compared to a violation of 39.5% resulting from the state-of-the-art cost minimization algorithm.

References

[1] A. Qureshi *et al.*, "Cutting the electric bill for internet-scale systems," in *SIGCOMM*, 2009.

[2] L. Rao *et al.*, "Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM*, 2010.

[3] F. Ahmad *et al.*, "Joint optimization of idle and cooling power in data centers while maintaining response time," in *ASPLOS*, 2010.

[4] B. Heller *et al.*, "Elastictree: Saving energy in data center networks," in *NSDI*, 2010.

[5] L. A. Barroso *et al.*, "The case for energy-proportional computing," *IEEE Computer*, pp. 33–37, December 2007.

[6] F. Li *et al.*, "Congestion and price prediction under load variation," *IEEE Transactions on Power Systems*, vol. 24, no. 2, pp. 911–922, 2009.

[7] R. Miller, "Who Has the Most Web Servers?" http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/, 2009.

[8] A. Greenberg *et al.*, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[9] A. Services, "Peak Power Rebate Overview," http://www.ameren.com.

[10] R. Raghavendra *et al.*, "No power struggles: Coordinated multi-level power management for the data center," in *ASPLOS*, 2008.

[11] X. Wang *et al.*, "Cluster-level feedback power control for performance optimization," in *HPCA*, 2008.

[12] X. Fan *et al.*, "Power provisioning for a warehouse-sized computer," in *ISCA*, 2007.

[13] F. Li, "Continuous locational marginal pricing (CLMP)," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1638 –1646, 2007.

[14] "ISO New England," http://www.iso-ne.com/.

[15] M. Colajanni *et al.*, "A performance study of robust load sharing strategies for distributed heterogeneous web server systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, 2002.

[16] J. M. others, "Cycle time approximations for the g/g/m queue subject to server failures and cycle time offsets with applications," in *ASMC*, 2006.

[17] G. Bolch *et al.*, *Queueing Networks and Markov Chains*. Wiley-Interscience, 2005.

[18] M. Al-Fares *et al.*, "A scalable, commodity data center network architecture," in *SIGCOMM*, 2008.

[19] J. Li *et al.*, "Towards optimal electric demand management for internet data centers," in *Techreport*, 2010.

[20] A. O. Allen, *Probability, Statistics, and Queuing Theory with Computer Science Applications*, 1990.

[21] G. Bolch *et al.*, *Probability, Statistics, and Queuing Theory with Computer Science Applications*, 1998.

[22] G. Trecate *et al.*, "Optimization with piecewise-affine cost functions," Technical Report AUT01-13, 2001, Tech. Rep.

[23] "Introduction to lp_solve 5.5.2.0," http://lpsolve.sourceforge.net/5.5/.

[24] "Data Center Knowledge," http://www.datacenterknowledge.com/, 2008.

[25] G. Urdaneta *et al.*, "Wikipedia workload analysis for decentralized hosting," *Elsevier Computer Networks*, vol. 53, no. 11, pp. 1830–1845, July 2009, http://www.globule.org/publi/WWADH_comnet2009.html.

[26] B. Urgaonkar *et al.*, "Dynamic provisioning of multi-tier internet applications," in *ICAC*, 2005.

[27] "PJM Training Materials LMP 101, PJM," http://pjm.com.

[28] G. Chen *et al.*, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *NSDI*, 2008.

[29] J. Chase *et al.*, "Managing energy and server resources in hosting centers," in *SOSP*, 2001.

[30] E. Elnozahy *et al.*, "Energy-efficient server clusters," *Power-Aware Computer Systems*, pp. 179–197, 2003.

[31] S. Nedevschi *et al.*, "Reducing network energy consumption via sleeping and rate-adaptation," in *NSDI*, 2008.

[32] K. Le *et al.*, "Managing the cost, energy consumption, and carbon footprint of internet services," in *SIGMETRICS*, 2010.

[33] K. Le, R. Bianchini *et al.*, "Capping the brown energy consumption of internet services at low cost," in *IGCC*, 2010.

[34] K. Le *et al.*, "Cost- and energy-aware load distribution across data centers," in *HOTPOWER*, 2009.

[35] Goiri *et al.*, "Intelligent placement of datacenters for internet services," in *ICDCS*, 2011.

[36] K. Le *et al.*, "Reducing electricity cost through virtual machine placement in high performance computing clouds," in *SC*, 2011.

[37] R. Urgaonkar *et al.*, "Optimal power cost management using stored energy in data centers," in *SIGMETRICS*, 2011.

[38] S. Govindan *et al.*, "Benefits and limitations of tapping into stored energy for datacenters," in *ISCA*, 2011.