12-2012

Power Management for Cloud-Scale Data Centers

Yanwei Zhang
yzhang82@utk.edu

To the Graduate Council:

I am submitting herewith a thesis written by Yanwei Zhang entitled "Power Management for Cloud-Scale Data Centers." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Xiaorui Wang, Major Professor

We have read this thesis and recommend its acceptance:

Gregory D. Peterson, Fangxing Li

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Power Management for Cloud-Scale Data Centers

A Thesis Presented for
The Master of Science
Degree
The University of Tennessee, Knoxville

Yanwei Zhang
December 2012

Acknowledgements

I would like to thank all the individuals who have encouraged, and inspired me in the preparation of my thesis. First and foremost, I would like to show my great gratitude towards my supervisor, Dr. Xiaorui Wang. His mentor has developed me the essential skills in solving different kinds of problems. I would also like to thank my special committee members for their time and input to my thesis: Dr. Gregory D. Peterson, and Dr. Fangxing Li.

I would also like to thank my supportive colleagues and friends at Univeristy of Tennessee for helping me in my study: Kai Ma, Xiaodong Wang, Xue Li, Chi Zhang, Ming Chen, Yanjun Yao, and Xing Fu. I also wish to give a special thank to my colleague, Yefu Wang, who is a close coworker on my thesis. His insight and discussion have done me a great favor.

At last, I owe my deepest gratitude to my dear parents and my dear older sister. Their constant support and love make me be a happy girl in pursuing my dream.

Abstract

Recent years have seen the rapid growth of large and geographically distributed data centers deployed by Internet service operators to support various services such as cloud computing. Consequently, high electricity bills, as well as negative environmental implications (*e.g.*, CO_2 emission and global warming) come along. In this thesis, we first 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 that impact the power markets. 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. We then propose *GreenWare*, a novel middleware system that conducts dynamic request dispatching to maximize the percentage of renewable energy used to power a network of distributed data centers, subject to the desired cost budget of the Internet service operator. Our solution first explicitly models the intermittent generation of renewable energy, e.g., wind power and solar power, with respect to varying weather conditions in the geographical location of each data center. We then formulate the core objective of GreenWare as a constrained

optimization problem and propose an efficient request dispatching algorithm based on linear-fractional programming (LFP).

Contents

List of Figures

Chapter 1

Introduction

Recent years have seen the rapid growth of large and geographically distributed data centers deployed by Internet service operators to support various services such as cloud computing. As an effort to deal with the increasingly severe global energy crisis, reducing the high energy consumption of those cloud-scale data centers has become a serious challenge. For example, some cloud-service data centers are termed as *mega data centers*, because they host hundreds of thousands of servers and can draw tens to hundreds of megawatts of power at peak [34]. It has also been reported that in a conservative estimation, Google hosts more than 500,000 servers in its data centers distributed in different locations and consumes at least 6.3×10^5 MWh in total annually [56]. Therefore, minimizing the energy consumption of cloud-scale data centers has recently received a lot of research attention (e.g., [29, 25, 42, 17, 67, 26]). However, much less attention has been given to a related but different research topic, *i.e.*, minimizing the electricity bill of a network of data centers by leveraging different electricity prices in different geographical location to wisely distribute workloads among those locations. Furthermore, in addition to high electricity bills, the enormous energy consumption of cloud-scale data centers can also lead to negative environmental implications (*e.g.*, CO_2 emission and global warming), due to their large carbon footprints. The reason is that most of the

produced electricity around the world comes from carbon-intensive approaches, *e.g.*, coal burning [42], in spite of some increasing efforts on promoting green energy generation. In particular, such energy produced with conventional fossil-based fuel is commonly referred to as brown energy; while in contrast, green (or clean) energy is normally generated from renewable energy sources, such as wind turbines and solar panels, and is thus more environmentally friendly.

Fortunately, the geographical distribution characteristic of the cloud-scale data centers often indicates a great chance in minimizing the electricity bill, as well as reducing the carbon footprints, for cloud-scale data center operators. This is due to the fact that data centers located in different regions often have some distinguishing qualities from each other, such as the variations shown in 1) the local power prices, and 2) the availabilities of the local renewable energy. To this end, this thesis puts an effort in proposing solutions to effectively reduce the electricity bill, as well as to green cloud-scale data centers for Internet service providers.

Electricity bill capping algorithm for cloud-scale data centers that impact the power markets

A few initial studies have been recently conducted to address the problem of electricity cost minimization [56, 58]. 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

[17, 35]. Recent studies show that cooling can take up to 25% [56] and network can account for up to 20% [35] 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 [20]. 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 [47]. 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 [50, 34] 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 the Ameren Illinois Utilities could save an average of 20% with the locational pricing policy [60]. 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 [34], 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 [57, 68]. 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 [30]. 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.

This thesis proposes 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.

A middleware system to maximize the use of renewable energy for cloud-scale data centers (GreenWare)

Solutions provided to manage the electricity bills in operating cloud-scale data centers can bring in significant economic gains for Internet service providers. However, they often follow by a zero renewable energy consumption, and thus a negative environmental implication. The reason is that currently renewable energy can be often more expensive to produce than brown energy [2, 15], due to the intermittent nature of renewable energy sources such as wind and sunlight. As a result, those solutions cannot be applied to mitigate the negative environmental implications caused by the rapidly increasing energy consumption in operating cloud-scale data centers. Therefore, in this work, we provide another solution to effectively green cloud-scale data centers, while controlling the electricity bills for Internet service providers.

Some attention has been paid on reducing brown energy consumption by cloud-scale data center operators. For example, major Internet service operators, *e.g.*, Google, Microsoft, and Yahoo!, have all started to increasingly power some of their data centers using renewable energy, and so reduce their dependence on brown energy [55, 4, 62]. Since data centers in different geographical locations may have different availabilities of renewable energy depending on the local weather conditions,

it is important for cloud-service operators to dynamically distribute service requests among different data centers to maximize the use of renewable energy.

Unfortunately, due to the intermittent nature of renewable energy sources such as wind and sunlight, currently renewable energy can be often more expensive to produce than brown energy [2, 15]. While some data centers are trying to build their own wind farms or solar photovoltaic (PV) power plants, due to concerns such as expensive facility investments and management, many Internet service operators choose to work with professional renewable energy producers and utilize the green energy integrated into the power grid. For example, Google has recently purchased 20 years' worth of wind energy from an Iowa wind farm, which will be sufficient to power several of its data centers in Oklahoma [16]. Google also invested $100 million in the Shepherds Flat Wind Farm in Oregon to generate 845 megawatts of green power, which will be sold directly to Southern California Edison's power grid. As a result of its higher production costs, renewable energy coming from the grid can be more expensive than brown energy. For example, the industrial electricity price for solar energy can be 16.14 cents per KWh in a sunny climate and 35.51 cents per KWh in a cloudy climate [11]. In contrast, the wholesale brown energy price can be around 6 cents per KWh [56]. The Los Angeles Department of Water and Power also estimates that the extra cost for green energy is at least 3 cents per KWh [7]. Therefore, utilizing renewable energy may impose a considerable pressure on the sometimes stringent operation budgets of Internet service operators, as the electricity cost of operating data centers has become a significant portion, *e.g.*, 20% or more of the monthly costs of those enterprises [34]. Hence, a key dilemma faced by many service operators is how to exploit renewable energy to the maximum degree that is allowed by their monthly operating budgets.

In this thesis, we then propose *GreenWare*, a novel middleware system that conducts dynamic request dispatching to maximize the percentage of renewable energy used to power a network of distributed data centers, subject to the desired cost budgets of Internet service operators. We first model the intermittent generation

of renewable energy, *i.e.*, wind power and solar power, with respect to the varying weather conditions in the geographical location of each data center. For example, the available wind power generated from wind turbines is modeled based on the ambient wind speed [52, 12], while the available solar power from solar plants is estimated by modeling the maximum power point on irradiance (*i.e.*, solar energy per unit area of the solar panel's face) and temperature [45, 59]. Based on the models, we formulate the core objective of GreenWare as a constrained optimization problem, in which the constraints capture the Quality of Service (QoS, *e.g.*, response time) requirements from customers, the intermittent availabilities of renewable energy in different locations, the peak power limit of each data center, and the monthly cost budget of the Internet service operator.

Contributions

Specifically, this thesis makes the following constributions.

For the electricity bill capping algorithm:

- 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.

For the GreeWare middleware system:

- We propose a novel GreenWare middleware system in operating geographically distributed cloud-scale data centers. GreenWare dynamically dispatches incoming service requests among different data centers, based on the time-varying electricity prices and availabilities of renewable energy in their geographical locations, to maximize the use of renewable energy, while enforcing the monthly budget determined by the Internet service operator.

- We explicitly model renewable energy generation, *i.e.*, wind turbines and solar panels, with respect to the varying weather conditions in the geographical location of each data center. As a result, our solution can effectively handle the intermittent supplies of renewable energy.

- We formulate the core objective of GreenWare as a constrained optimization problem and propose an efficient request dispatching solution based on LFP.

- We evaluate GreenWare with real-world weather, electricity price, and workload traces. Our experimental results show that GreenWare can significantly reduce the dependence of cloud-scale data centers on fossil-fuel-based energy without violating the desired cost budget, despite the intermittent supplies of renewable energy and time-varying electricity prices and workloads.

The rest of this thesis is organized as follows. Chapter 2 discusses the related work. Chapter 3 proposes an electricity bill capping algorithm for cloud-scale data centers that impact the power markets. Chapter 4 proposes a middleware system that conducts dynamic request dispatching to maximize the percentage of renewable energy used to power a network of distributed data centers, subject to the desired cost budget. Chapter 5 concluses this thesis.

Chapter 2

Related Work

Electricity bill and carbon footprints are both important concerns for Internet service providers. Research works to the topic in this thesis fall into three categories.

2.1 Conserving Energy Consumption

Many recent research projects have tried to minimize the energy consumption of data centers. For example, Chen et al. [26] and Chase et al. [25] reduce the energy consumption of connection servers hosting long-lived TCP-connection services and web servers providing request-response type of services, respectively. Heo et al. [36] have developed an adaptation graph analysis mechanism to solve the conflicts between interacting adaptive components, *e.g.*, *On/Off* and *dynamic voltage scaling* policies in server farms, to minimize energy consumption. Elnozahy et al. [29] investigate various combinations of dynamic voltage scaling and node on/off policies to reduce the energy consumption in server farms. Other strategies on reducing energy consumption of servers are also proposed (*e.g.*, [39, 67]).

However, none of the aforementioned works are designed to directly lower the electricity bill for Internet service operators, and none of them try to utilize renewable energy to power data center networks.

2.2 Managing Electricity Cost

A few recent projects have proposed to minimize the electricity bills of data center networks. For example, Qureshi et al. [56] try to lower the electricity bill by utilizing the varying electricity prices in different locations of distributed data centers. Rao et al. [58] consider a multi-electricity-market environment to reduce the electricity bill. In a recent study, Zhang et al. [69] propose an electricity bill capping algorithm to minimize the electricity cost within the cost budget for data center networks. Lin et al. [49] have tried to minimize the energy cost together with delay cost by rightly sizing data centers. In [32], 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 [43], Le et al. investigate policies for virtual machine migration across data center networks, to lower electricity costs. In addition, Urgaonkar et al [66] and Govindan et al [33] 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. Furthermore, none of them try to maximize the use of renewable energy in powering data center networks for the Internet service operators.

2.3 Utilizing Renewable Energy

This is a relatively new topic with only few initial studies. Le et al. [42, 41] propose to cap the consumption of brown energy while maintaining service level agreements (SLAs). Liu et al. [66] investigate how renewable energy can be used to lower the

electricity price of brown energy in a specific power market, *i.e.*, where the brown energy is dynamically priced in proportion to the total brown energy consumption. Brown et al. [23] propose a simulation infrastructure to model a data center using renewable energy sources. In contrast to those studies, GreenWare aims to solve a related but different problem, *i.e.*, maximizing the use of renewable energy subject to the cost budget of the Internet service operators. Steward et al. [61] also try to maximize the use of renewable energy in data centers. However, their study assumes that Internet service operators have their own wind farms or solar plants. In contrast, GreenWare considers a different case where the service operators buy renewable energy from the power grid, which is a more common case for many data centers because of concerns such as expensive facility investments and management. In addition, their study does not consider the extra cost of renewable energy and may lead to budget violations. Li et al. [44] propose a load power tuning scheme for managing intermittent renewable power in a single data center without considering the costs. In contrast, we focus on distributing requests among data centers in different locations.

Chapter 3

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

As discussed in Chapter 1, to minimize the electricity bill of a network of data centers by leveraging different electricity prices in different geographical locations to distribute workloads among those locations, the initial solutions 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 chapter, 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.

3.1 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 [46]. 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 [46], 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 [47, 46]. 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 with the power demands in the local power markets [6].

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 3.1 shows the locational pricing policies in the three locations of B, C, and D from the well-known PJM five-bus system [46], 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 [47]. 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 3.1 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 [50, 34]. 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 [56, 58] 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 3.3.

3.2 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.

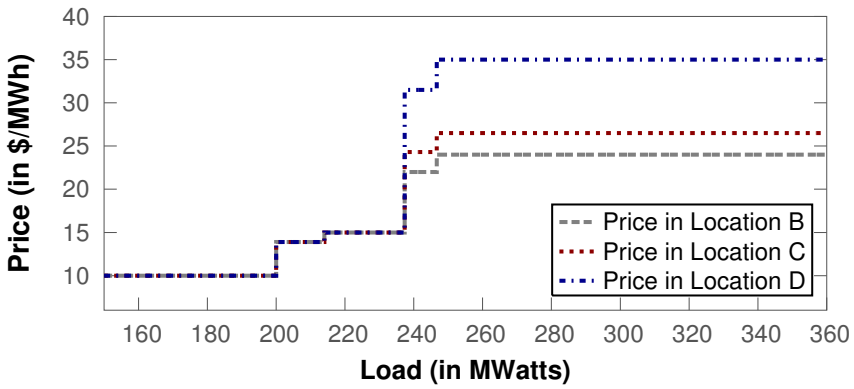

Figure 3.1: Locational electricity pricing policies in three locations in the state of New York.

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 [46].

As shown in Figure 3.2, 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 [17] for data center-level management algorithms.

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

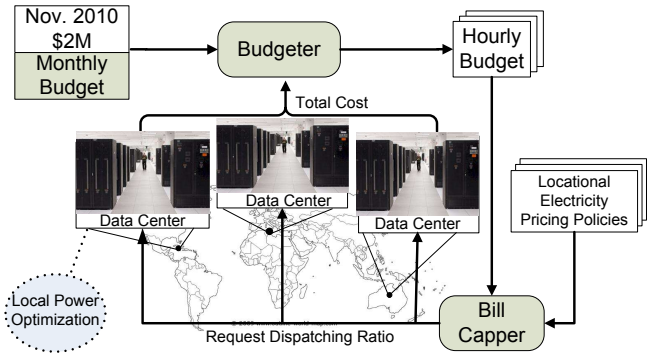

Figure 3.2: Proposed electricity cost capping architecture for distributed cloud-scale data centers.

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 3.5.2. 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 [56]. 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 [27]. 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 3.3.2, 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.

3.3 Cost Minimization

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

3.3.1 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

$$Minimize : \sum_{i=1}^{N} Pr_i \cdot p_i \tag{3.1}$$

such that

$$(a) \sum_{i=1}^{N} \lambda_i = \lambda; (b) p_i \leq Ps_i; (c) R_i \leq Rs_i \tag{3.2}$$

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 (3.1 - 3.2), 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 [17, 51, 22, 18, 35, 48]. Furthermore, without loss of generality, our framework can be easily integrated with more detailed and sophisticated power models.

3.3.2 Performance and Power Models

Queueing theory is commonly used in modeling system performance, such as in [17, 51, 22]. In this paper, we model a data center as a G/G/m queue [17] 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 [19, 21] in modeling the G/G/m queue, we have

$$R_i = \frac{1}{\mu_i} + \frac{C_A^2 + C_B^2}{2} \cdot \frac{(\rho^{n_i} + \rho)/2}{n_i \cdot \mu_i - \lambda_i} \tag{3.3}$$

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 (3.3), 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 (3.3) 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 (3.3) 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 [58].

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 [17].

$$p_i = p_i^{server} + p_i^{networking} + p_i^{cooling} \tag{3.4}$$

Power model for servers. As indicated in Figure 3.2, 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 (3.3). The power consumed by all the active servers in the data center is then modeled as

$$p_i^{server} = \sum_{k=1}^{n_i} sp_i^k \tag{3.5}$$

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 [17]. 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 (3.5) 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 (3.5) 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 [35]. 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

$$p_i^{networking} = A_i \cdot esp_i + B_i \cdot asp_i + C_i \cdot csp_i \tag{3.6}$$

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% [35]. Thus, the power consumption of networking devices is a function of the number of active switches, which vary significantly based on data center workloads [35].

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 [48] 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. [17].

$$p_i^{cooling} = coe_i^{-1} \cdot (p_i^{server} + p_i^{networking}) \tag{3.7}$$

3.3.3 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) - (3.7) is non-linear since the pricing policies involved in $Pr_i = F(p_i + d_i)$ are usually non-linear [46, 47]. As discussed in Section 3.1, 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 [63] to formulate the problems in (1) - (3.9) 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 [63]. After the transformation, a standard MILP solver (*e.g.*, lp_solver) [5], 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 [14]. 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 [47]. Based on the simulation in Section 3.5, 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.

3.4 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 3.3, 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:

$$Maximize : \sum_{i=1}^{N} \lambda_i \tag{3.8}$$

such that $$(a)\sum_{i=1}^{N} C_i \leq Cs; (b)p_i \leq Ps_i; (c)R_i \leq Rs_i \tag{3.9}$$

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.

3.4.1 Sufficient Cost Budget

The objective of the optimization problem in (3.8) 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 (3.8). 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.

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

3.4.2 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 (3.8) will be reconfigured as a cost minimization problem in the form of (3.1 - 3.2) 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.

3.5 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 [58, 17]. 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.

3.5.1 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 [50]. 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 [58]. Specifically, the server configuration in each location is respectively assumed to be as follows [48]: 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 [35].

3.5.2 Real-World Traces

To build our workloads in the simulator, we use a trace of Internet traffic from Wikipedia.org [64]. 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 [65], 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 [9]. Additionally, in order to simulate the cooling strategy run in data centers discussed in Section 3.3.2, we refer to the cooling efficiencies as, 1.94, 1.39, and 1.74 for the three data centers [48], respectively. The pricing policies used in our simulation are generated based on the well-known PJM five-bus system as Figure 3.1.

3.6 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.

3.6.1 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 [58]. 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 3.1. 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 3.1.

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.

3.6.2 Electricity Cost Minimization

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

Figure 3.3 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 3.1, the key

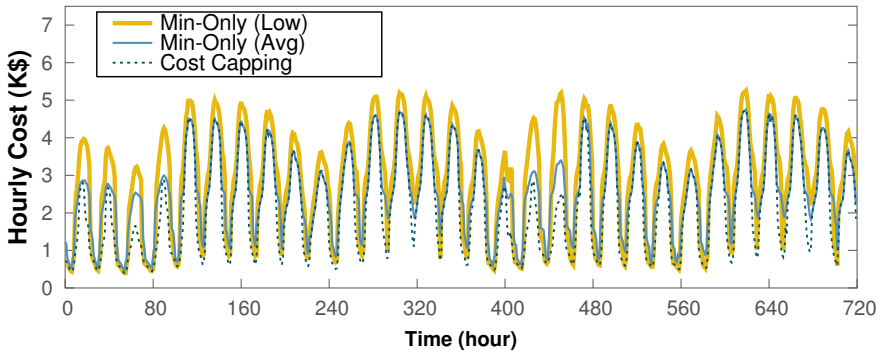

Figure 3.3: Hourly electricity cost comparison between Cost Capping and Min-Only with respect to Nov. 2007 Wikipedia trace.

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 3.4 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 3.1; 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 3.4 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

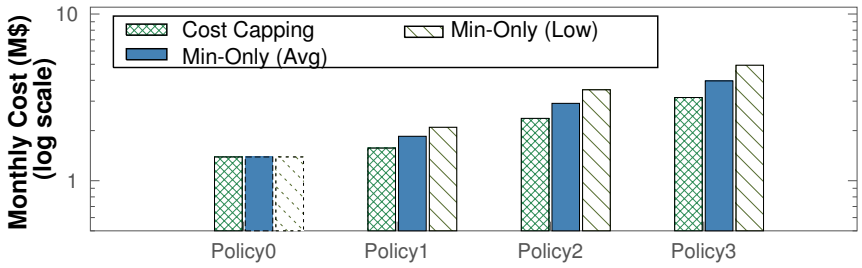

Figure 3.4: Monthly electricity bills comparison under different pricing policies with respect to Nov. 2007 Wikipedia trace.

Min-Only, due to the fact that it considers the locational pricing policies in the real-world 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 3.3.

3.6.3 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 3.4. 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

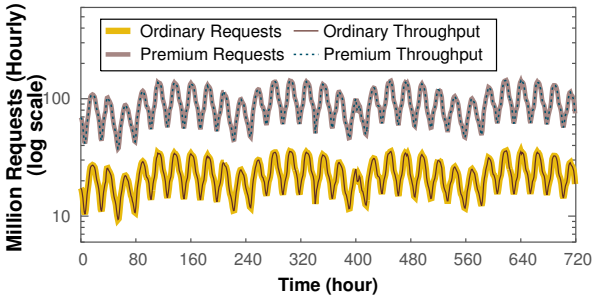

Figure 3.5: Throughput by Cost Capping under a monthly cost budget of $\$2.5M$ with respect to Nov. 2007 Wikipedia trace.

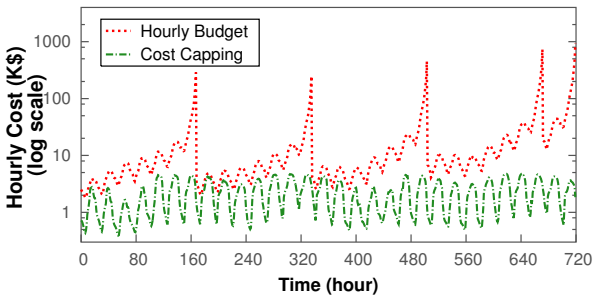

Figure 3.6: Hourly electricity cost capping by Cost Capping under a monthly cost budget of $\$2.5M$ with respect to Nov. 2007 Wikipedia trace.

a monthly cost budget of $\$2.5M$, which is estimated based on the workload in Nov. 2007 from the Wikipedia website.

Figures 3.5 and 3.6 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 3.5; and 2) The electricity cost in each hour is below the given cost budget, as shown in Figure 3.6. In addition, Figure 3.6 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 un-used allocated cost budget from previous invocation periods to the remaining invocation periods in the same week.

As shown in Figures 3.7 and 3.8, 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

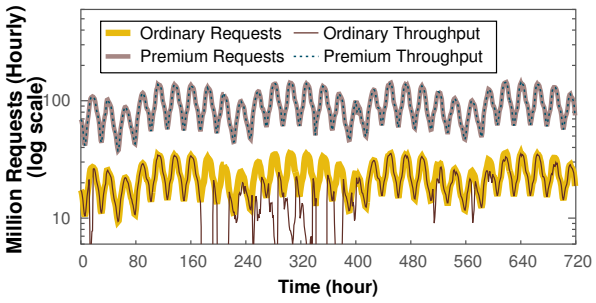

Figure 3.7: Throughput by Cost Capping under a monthly cost budget of $\$1.5M$ with respect to Nov. 2007 Wikipedia trace.

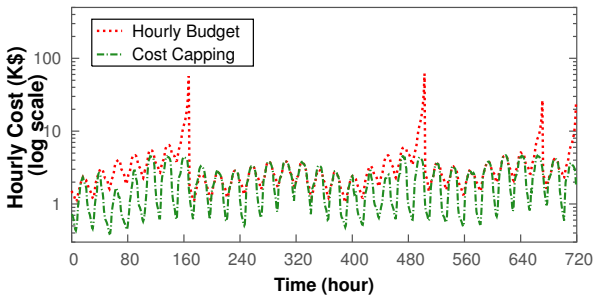

Figure 3.8: Hourly electricity cost capping by Cost Capping under a monthly cost budget of $\$1.5M$ with respect to Nov. 2007 Wikipedia trace.

requests are serviced, *e.g.*, the hours of 176, 177, 178, 202 in Figure 3.7, 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 3.8. Second, for those invocation periods where certain ordinary requests are serviced, *e.g.*, the hours of 13, 14, 15 in Figure 3.7, 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 3.9 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 3.9 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 3.10 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 3.10, 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.

3.7 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

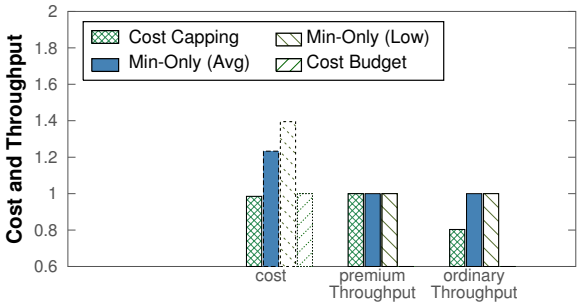

Figure 3.9: Cost and throughput comparisons under a monthly cost budget of $\$1.5M$ with respect to Nov. 2007 Wikipedia trace.

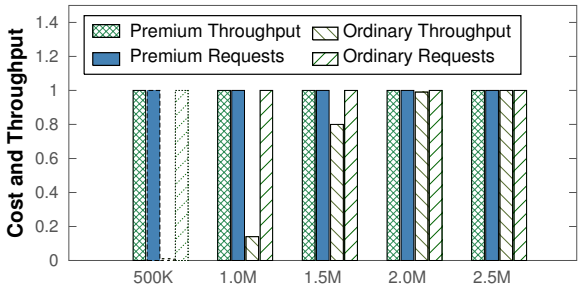

Figure 3.10: Monthly throughput by Cost Capping with a series of monthly cost budgets with respect to Nov. 2007 Wikipedia trace.

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 [58, 56], 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.

3.8 Summary

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 chapter, 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.

Chapter 4

GreenWare: Greening Cloud-Scale Data Centers to Maximize the Use of Renewable Energy

As discussed in Chapter 1, the electricity bill capping algorithm in Chapter 3 can not only minimize the electricity bill, but also enforce a cost budget on the monthly bill for cloud-scale data centers that impact the power markets. However, those solutions often follow by a zero renewable energy consumption, and thus a negative environmental implication. The reason is that currently renewable energy can be often more expensive to produce than brown energy [2, 15], due to the intermittent nature of renewable energy sources such as wind and sunlight. As a result, those solutions cannot be applied to mitigate the negative environmental implications caused by the rapidly increasing energy consumption in operating cloud-scale data centers. This chapter proposes *GreenWare*, a novel middleware system that conducts dynamic request dispatching to maximize the percentage of renewable energy used to power a network of distributed data centers, subject to the desired cost budgets of Internet service operators. We first model the intermittent generation of renewable energy, *i.e.*, wind power and solar power, with respect to the varying

weather conditions in the geographical location of each data center. For example, the available wind power generated from wind turbines is modeled based on the ambient wind speed [52, 12], while the available solar power from solar plants is estimated by modeling the maximum power point on irradiance (*i.e.*, solar energy per unit area of the solar panel's face) and temperature [45, 59]. Based on the models, we formulate the core objective of GreenWare as a constrained optimization problem, in which the constraints capture the Quality of Service (QoS, *e.g.*, response time) requirements from customers, the intermittent availabilities of renewable energy in different locations, the peak power limit of each data center, and the monthly cost budget of the Internet service operator. We then transfer the optimization problem into a linear-fractional programming (LFP) formulation for an efficient request dispatching solution with a polynomial time average complexity.

4.1 GreenWare Architecture

In this section, we provide a high-level description of the proposed GreenWare system. GreenWare dynamically conducts request dispatching among data centers in order to maximize the percentage of renewable energy used to power a network of distributed data centers, based on the time-varying electricity prices and availabilities of renewable energy in their geographical locations. In the meantime, GreenWare guarantees the desired QoS for customers and effectively maintains the electricity bill within a cost budget determined by the Internet service operators.

In this work, we assume that a network of distributed data centers share a common cost budget, which can be determined by the Internet service operator periodically in each budgeting period (*e.g.*, a month). A local optimizer is assumed to be present in each single data center in the network to dynamically adjust the number of active servers to minimize the power consumption of the data center, while maintaining a desired level of QoS based on a QoS model detailed in Section 4.2.2. We also assume that the short-term weather conditions (*e.g.*, in one hour) and the configurations

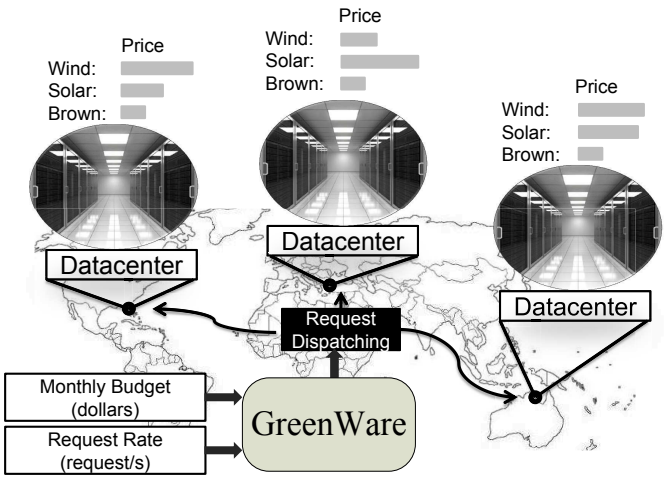

Figure 4.1: Proposed GreenWare system for distributed cloud-scale data center networks.

of wind turbines and solar panels of each data center are available. As shown in Figure 4.1, *GreenWare* is a centralized system that manages a data center network for maximizing the use of renewable energy within the cost budget. While such a centralized architecture is commonly used in the management of data center networks [42, 58, 56], GreenWare can be extended to work in a hierarchical way, which is our future work. Similar to [17, 66, 69], we use one month as the budgeting period and one hour as the period for GreenWare to be invoked and conduct the request dispatching operation.

In every invocation period, GreenWare performs three steps: First, GreenWare computes the hourly budget based on the monthly cost budget from the service operator 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 4.3.3. Second, based on the time-varying electricity prices and availability of renewable energy at each data center, with respect to the varying weather conditions in their geographical location (*e.g.*, irradiance, temperature, and wind speed), GreenWare runs the optimization algorithm in Section 4.2 to compute the desired request dispatching (*e.g.*, the fraction of workload allocated to each data center) such that (1) the overall percentage of

renewable energy used to power a network of distributed data centers is maximized within budget constraints; (2) the total electricity cost is below the budget of the current hour; and (3) the application-level QoS (*e.g.*, desired response time) for customers is guaranteed. Third, GreenWare redirects the incoming requests among data centers based on the determined request dispatching in Step (2), using the dynamic request routing mechanism already deployed in cloud-scale data center networks. Note that dynamic request routing has already been implemented by many Internet service operators to map requests to servers, for the purposes of customer QoS guarantees and fault-tolerance [56].

4.2 Design Methodology of GreenWare

In this section, we first present the problem formulation of the optimization objective of GreenWare. We then introduce the adopted performance and server power models, as well as the wind power model and solar power model. Finally, we discuss our request dispatching solution. Note that we focus mainly on wind power and solar power in this work because there exists meteorological data [8] for us to simulate their intermittent availabilities in distributed data centers. GreenWare can be applied to other types of renewable energy, such as hydro-electric and geothermal, if their corresponding meteorological data is also available.

4.2.1 Problem Formulation

We first introduce the following notation. N data centers are operated in a cloud-scale data-center network. The i^{th} data center consumes pW_i kilowatts of wind energy, pS_i kilowatts of solar energy and pB_i kilowatts of brown energy, respectively. The total power consumption p_i (*i.e.*, $p_i = pW_i + pS_i + pB_i$) of the i^{th} data center should not exceed the peak power limit Ps_i of the data center. The intermittent availabilities of the renewable energy in the local power market of the i^{th} data center are denoted as

PW_i and PS_i. In particular, PW_i and PS_i are the estimated wind power output from the wind farm and the maximum solar power output from the solar plant, respectively. The corresponding wind farm and solar plant are assumed to be the renewable energy sources for the local power market of the i^{th} data center. PrW_i, PrW_i and PrB_i are the current electricity prices of the three types of energy from the power market of the i^{th} data center, respectively. The whole system has an incoming workload of λ requests per hour. Our algorithm allocates the i^{th} data center with a workload of λ_i requests per hour to maximize the percentage of renewable energy used, depending on the wind and solar power models based on local weather conditions (presented in Sections 4.2.3 and 4.2.4), within the allocated cost budget Cs. The average response time of the i^{th} data center is R_i and the corresponding response time set point is Rs_i.

Given a workload of λ requests per hour, the optimization goal is to dynamically choose a request dispatching strategy such that the i^{th} data center is assigned λ_i requests to maximally use renewable energy to power the data center network within the cost budget. Specifically, in order to maximize the overall renewable energy usage of all the N data centers, x_i percentage of wind power and y_i percentage of solar power out of the total power consumption p_i by the i^{th} data center will have to be determined. Then, z_i percentage of the total power consumption is supplemented in the form of brown energy. It is clear that $z_i = 1 - x_i - y_i$. In summary, the optimization problem can be expressed as follows.

Problem 1:

$$Maximize : \frac{\sum_{i=1}^{N} (pW_i + pS_i)}{\sum_{i=1}^{N} (pW_i + pS_i + pB_i)} \tag{4.1}$$

subject to

$$\sum_{i=1}^{N} \lambda_i = \lambda \tag{4.2}$$

$$\lambda_i \geq 0 \tag{4.3}$$

$$R_i \leq Rs_i \tag{4.4}$$

$$0 \leq pW_i \leq PW_i \tag{4.5}$$

$$0 \leq pS_i \leq PS_i \tag{4.6}$$

$$0 \leq pW_i + pS_i + pB_i \leq Ps_i \tag{4.7}$$

$$\sum_{i=1}^{N}(PrW_i \cdot pW_i + PrS_i \cdot pS_i + PrB_i \cdot pB_i) \leq Cs \tag{4.8}$$

Specifically, pW_i, pS_i, pB_i, PW_i, and PS_i (in KW) will be numerically the same as energy (in KWh) since the invocation period used in this work is assumed to be one hour. In order to solve the optimization **Problem 1**, it is important to model the variables pW_i, pS_i and pB_i as functions of λ_i, x_i and y_i. It is clear that

$$pW_i = x_i \cdot p_i; pS_i = y_i \cdot p_i; pB_i = z_i \cdot p_i \tag{4.9}$$

where $x_i + y_i + z_i = 1$.

Thus, in the following we first model the power consumption p_i and the average response time R_i with the request distribution rate λ_i for the i^{th} data center. We then model the availabilities of wind power and solar power, *i.e.*, PW_i and PS_i, respectively, based on the weather condition of the i^{th} data center, *e.g.*, irradiance, temperature, and wind velocity. We discuss an efficient solution design for **Problem 1** in Section 4.2.5.

4.2.2 Response Time and Power Models

Queueing theory is widely used to model the performance of a web server [17, 22]. In this paper, we use the M/M/n queueing model in queueing theory [58] to model the response time for a data center. The average response time of the requests to a web server consists of two portions: (1) the average waiting time that the requests spend in a queue waiting to be serviced and (2) the service time, *i.e.*, $\frac{1}{\mu}$, given the service

rate μ of the data center. Specifically, the average waiting time for a data center with n active servers can be expressed as $\frac{1}{n \cdot \mu - \lambda} \cdot P_Q$, where P_Q represents the probability that the incoming requests need to wait in a queue to be serviced. Furthermore, we assume that all the active servers will likely keep busy, *i.e.*, running at close to 100% utilization, because a local optimizer running in each data center minimizes the number of active servers. Hence, without loss of generality, P_Q is assumed to be 1, since all the active servers are assumed to be running at close to 100% utilization. The same assumption is used in existing solutions on electricity cost minimization for data centers [58]. Therefore, we have

$$R_i = \frac{1}{\mu_i} + \frac{1}{n_i \cdot \mu_i - \lambda_i} \tag{4.10}$$

where n_i is the number of active servers and μ_i is the average service rate of a single server, *i.e.*, the number of requests the server is able to process in a unit time, in the i^{th} data center .

As discussed in Section 4.1, we assume that a local optimizer runs in every data center and dynamically adjusts the number of active servers to provide a desired level of QoS (*e.g.*, response time) with the least number of servers. As a result, given a request rate of λ_i and a desired response time Rs_i of the i^{th} data center, the number of desired active servers n_i can be derived from equation (4.10). Thus, we have $p_i = n_i \cdot sp_i$, where sp_i is the average power consumption of a single server in the i^{th} data center. Although the power consumption of a server is usually a function of the utilization of the server, we assume that sp_i is constant because when the local optimizer minimizes the number of active servers, all the servers remaining active will likely run close to a 100% utilization. Thus, the utilization will be approximately the same. It is then clear that a linear server power model based on the incoming work rate λ_i for the i^{th} data center can be derived, *i.e.*, $p_i = f(\lambda_i)$, where $f(\lambda_i)$ is a linear function.

4.2.3 Wind Power Model

The number of wind turbine installations is rapidly growing worldwide. It is expected that the US can get 20% of its electricity from wind energy by the year 2030 [38, 54]. It has been shown that wind power generated by wind turbines in a wind farm can be modeled as a function of the actual wind speed [52, 12]. For example, the wind power output p_{wind} by a single wind turbine, with respect to a wind speed of v, can be approximated as follows

$$p_{wind} = \begin{cases} 0 & v < v_{in}, v > v_{out} \\ p_r \cdot \frac{v - v_{in}}{v_r - v_{in}} & v_{in} < v < v_r \\ p_r & v_r < v < v_{out} \end{cases}$$

where v_r, p_r are the rated speed and power of the wind turbine and v_{in}, v_{out} are cut-in and cut-out wind speeds. Specifically, the cut-in speed is the wind speed at which the turbine first starts to rotate and generate power, *e.g.*, a typical value between 3 and 4 meters per second; while the cut-out speed is employed by the braking system to bring the rotor to a standstill to eliminate the risk of damaging the turbine rotor due to the continuously rising wind speed, *e.g.*, a cut-out speed of usually around 25 meters per second.

In the case of a large-scale wind power generation farm, *e.g.*, one consisting of a large number m_w of wind-turbines, the overall wind power output is estimated as the sum of the power output values sampled at different turbines for simplicity [31]

$$PW = \sum_{k=1}^{m_w} p_{wind}^k$$

where p_{wind}^k is the power output from the k^{th} wind turbine with respect to the wind speed v, with t05e assumption that the wind turbines have the same wind speed in the same wind farm.

4.2.4 Solar Power Model

The worldwide photovoltaic (PV) power capacity installation grows in a nearly exponential way, despite their relatively high cost [59]. In this work, we model the solar power generated by solar plants with respect to the varying weather conditions, such as irradiance and temperature, based on a single diode equation [59, 53]. In particular, the single diode equation has been widely used to simulate the available electrical power generated from a single PV panel. Specifically, the resulting current-voltage characteristic of a PV panel is

$$i = I_{ph} - I_o \cdot (e^{\frac{v+i \cdot R_s}{n_s \cdot V_{th}}} - 1) - \frac{v + i \cdot R_s}{R_{sh}} \tag{4.11}$$

where I_{ph} is the photo-generated current while I_o is the dark saturation current with respect to the ambient weather pattern. Moreover, the single-diode model takes into account both the series and parallel (shunt) resistance of the PV panel, referred to as R_s and R_{sh}, respectively. V_{th} is the junction thermal voltage, *i.e.*, $V_{th} = k \cdot T/q$, where k is Boltzmann's constant, q is the charge of the electron and T is the ambient temperature. n_s is the number of the solar cells in the PV panel connected in series, *e.g.*, $n_s = 72$ in BP-MSX 120 panels [1].

To show the solar power output from PV panels with respect to the varying weather conditions (*e.g.*, irradiance and temperature), equation (4.11) can then be transformed as equation (4.12) by including these two key factors, *i.e.*, irradiance and temperature [59]. In particular, it has been demonstrated that the dark saturation current of I_o just varies with the ambient temperature T, independent on the irradiance condition G [59, 24]. Furthermore, for a high-quality solar cell, it typically has a low series resistance R_s but a high parallel resistance R_{sh}. As a result, the solar model in this work only takes into considerations the series resistance (*i.e.*, $R_{sh} = \infty$), which is consistent with the prior study [45]. We thus have the fact that I_{ph} can be approximated by I_{sc} for simplicity, where I_{sc} is the short-circuit current. In particular, I_{sc} is directly proportional to the irradiance as well as the ambient

temperature. Thus, we have

$$i(G,T) = I_{sc}(G,T) - I_o(T) \cdot e^{\frac{v(G,T)+i(G,T)\cdot R_s}{n_s \cdot V_{th}}} \tag{4.12}$$

where $I_{sc}(G,T) = \frac{G}{G_0} \cdot I_{sc} \cdot (1 + \frac{k_i}{100} \cdot (T - T_0))$ and $I_o(T) = I_{sc} \cdot (1 + \frac{k_i}{100} \cdot (T - T_0)) \cdot e^{-\frac{V_{oc}+k_v \cdot (T-T_0)}{n_s \cdot V_{th}}}$. G_0 and T_0 are the respective irradiance level and temperature in Standard Test Conditions (STC), *i.e.*, $G_0 = 1000W/m^2$ and $T_0 = 25^oC$. I_{sc}, V_{oc}, k_v and k_i are the given parameters of short-circuit current, open-circuit voltage, temperature coefficients of the short-circuit and the open-circuit in STC from the datasheet of PV panels, respectively.

In particular, the solar power produced by a PV panel with respect to the varying weather conditions, based on the current-voltage characteristic shown as equation (4.11) is the product of the output voltage and current. Namely, $p_{solar} = v(G,T) \cdot i(G,T)$. It has been demonstrated that the power output p_{solar} generated by a PV panel shows a unique maximum value under uniform irradiation and temperature [45, 59]. In order to achieve the maximum efficiency of solar plants, some researchers have already put efforts in extracting the maximum power point from solar plants [28, 40]. We thus estimate the solar power output by a PV panel as the maximal power value which can be extracted from the PV panel (referred to as *mpp*). Specifically, *mpp* is achieved with respect to an optimal load r_{mp} and the corresponding current i_{mp} [28], where $r_{mp} = R_s + \frac{n_s \cdot V_{th}}{I_{sc}(G,T)+I_o(T)-i_{mp}}$. Thus, $mpp = i_{mp}^2 \cdot r_{mp}$. The Lambert W-function method is then used to calculate the maximum power point *mpp* of the PV panel with respect to the varying weather conditions. We assume that there are m_s PV panels installed in a large-scale solar plant. Thus, the overall solar power output by the solar plant is estimated as

$$PS = \sum_{k=1}^{m_s} mpp^k$$

where mpp^k is the maximum power point from the k^{th} PV panel with respect to the irradiance G and temperature T.

4.2.5 Problem Solution

Based on the analysis above, the optimization **Problem 1** is a non-linear programming problem with both a non-linear objective function and non-linear constraints, with respect to decision variables of λ_i, x_i and y_i. However, for a service operator, it is important to design an efficient solution in order to dynamically make decisions to green the data centers with acceptable runtime overheads. We thus transfer the non-linear optimization **Problem 1** into a well-studied linear-fractional programming formulation as in the form of **Problem 2**, which can be further transferred into a standard linear programming problem. Specifically, note that for the equations (4.9) with respect to pW_i, pS_i and pB_i as discussed in Section 4.2.1, we can alternatively assume that among the λ_i requests serviced by the i^{th} data center, λ_i^W, λ_i^S and λ_i^B requests are serviced with wind energy, solar energy and brown energy, respectively. Thus, we can limit the decision variables for the optimization **Problem 1** in (4.1 - 4.8) to only workload-related variables of λ_i^W, λ_i^S and λ_i^B, instead of both workload-related variables (*i.e.*, λ_i) and percentage variables (*i.e.*, x_i and y_i).

Since $\lambda_i = \lambda_i^W + \lambda_i^S + \lambda_i^B$, **Problem 1** in (4.1 - 4.8) can be further transferred as follows.

Problem 2:

$$Maximize : \frac{\sum_{i=1}^{N} f(\lambda_i^W + \lambda_i^S)}{\sum_{i=1}^{N} f(\lambda_i^W + \lambda_i^S + \lambda_i^B)} \tag{4.13}$$

subject to

$$\sum_{i=1}^{N} (\lambda_i^W + \lambda_i^S + \lambda_i^B) = \lambda \tag{4.14}$$

$$\lambda_i^W \geq 0 \tag{4.15}$$

$$\lambda_i^S \geq 0 \tag{4.16}$$

$$\lambda_i^B \geq 0 \tag{4.17}$$

$$R_i \leq Rs_i \tag{4.18}$$

$$0 \leq f(\lambda_i^W) \leq PW_i \tag{4.19}$$

$$0 \leq f(\lambda_i^S) \leq PS_i \tag{4.20}$$

$$0 \leq f(\lambda_i^W + \lambda_i^S + \lambda_i^B) \leq Ps_i \tag{4.21}$$

$$\sum_{i=1}^{N}(PrW_i \cdot f(\lambda_i^W) + PrS_i \cdot f(\lambda_i^S) + PrB_i \cdot f(\lambda_i^B)) \leq Cs \tag{4.22}$$

Specifically, $f(\lambda_i^W)$, $f(\lambda_i^S)$ and $f(\lambda_i^B)$ represent the amount of wind energy, solar energy and brown energy consumed in the i^{th} data center, respectively. It is clear that $f(\lambda_i^W)$, $f(\lambda_i^S)$ and $f(\lambda_i^B)$ are linear functions as discussed in Section 4.2.2.

Problem 2 is thus a specific case of linear-fractional programming problem with a fractional objective function and linear constraints. In order to solve the LFP-based optimization **Problem 2**, we leverage a standard technique discussed in [37] to transfer the problem in (4.13 - 4.22) to a linear programming problem. The detailed transformation is not shown due to space limitations, but the steps can be found in [37]. In our system, we implement the proposed GreenWare middleware system based on the *linprog* solver in Matlab. In particular, *linprog* uses an simplex method, which has been proven to have a low complexity in practice [10].

4.3 Simulation Setup

We aim to use realistic parameters in our experimental setup. We design a simulator and use real-world weather data, Web request traces, as well as electricity price data from utility companies to evaluate the proposed GreenWare system. As discussed, GreenWare dynamically conducts request dispatching to maximize the percentage of renewable energy used to power a network of distributed data centers within the

cost budget determined by the Internet service operator. These evaluations primarily target web server-based applications, which provide the request-response type of web services. Specifically, the setup simulates an Internet-scale data center network such as Google's data centers within the US.

4.3.1 Datacenter Parameters

In our evaluation, we simulate a large system composed of four geographically distributed data centers for an Internet service operator (*e.g.*, Google). Accordingly, four different locations are assumed in the simulator, *i.e.*, *San Luis Valley* in *Colorado*, *Los Angeles* in *California*, *Oak Ridge* in *Tennessee* and *Lanai* in *Hawaii*, which are the locations whose meteorological data are available in [8].

The power consumption profile of each server in the same location is assumed to be approximately the same, which is usually true when homogeneous servers and configurations are used in each data center [58, 49]. Specifically, similar to a related study [48], the server configuration in each location is respectively assumed to be as follows: Data Center 1 (2.0 GHz AMD Athlon processor), Data Center 2 (1.2 GHz Intel Pentium 4630 processor), Data Center 3 (2.9 GHz Intel Pentium D950 processor), and Data Center 4 (2.7 GHz AMD Athlon processor). Their power consumption is assumed to be 88.88, 34.10, 149.19, and 141.28 Watts and their processing capacity coefficients are estimated as 500, 300, 725, and 675 requests per second, respectively.

4.3.2 Renewable Energy Availability

To emulate the intermittent availabilities of renewable energy in the locations of different data centers, *i.e.*, wind power and solar power, we use meteorological data from the Measurement and Instrumentation Data Center (MIDC) [8] of the National Renewable Energy Laboratory. A variety of meteorological data, including irradiances, temperature, and wind speed, is covered in those records from the MIDC.

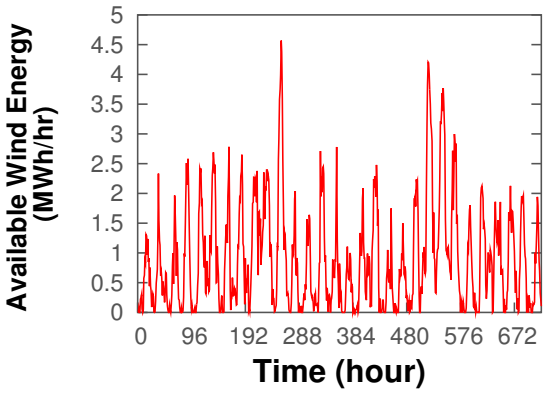

Figure 4.2: The trace of available wind energy throughout the entire simulated month.

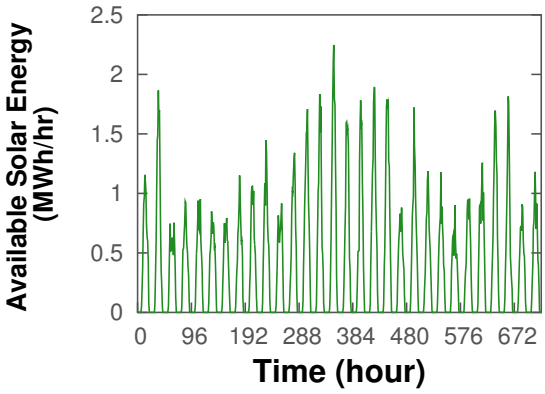

Figure 4.3: The trace of available solar energy throughout the entire simulated month.

Moreover, prior studies have shown that the data from the MIDC is sufficiently accurate [45]. In particular, we use meteorological data from the four stations, *e.g.*, *Sun Spot One*, *Loyola Marymount University Rotating Shadowband Radiometer*, *Oak Ridge National Laboratory* and *La Ola Lanai*, since they have consistent time periods with available meteorological data, beginning from June 1st, 2010 to June 30th, 2010. We further assume that there are 200 turbines installed in each wind farm and 10,000 solar panels installed in each solar plant to provide renewable energy to the local power utilities of the 4 data centers. In particular, BP-MSX 120 panels produced by British Petroleum are assumed to be used in the solar plants [1].

Specifically, based on the power models discussed in Sections 4.2.3 and 4.2.4, as well as the varying weather conditions obtained from MIDC, the available renewable energy of all the 4 data centers throughout the entire simulated month is demonstrated in Figures 4.2 and 4.3. In particular, Figure 4.2 depicts the overall available wind energy of all the 4 data centers, while Figure 4.3 shows the overall available solar energy. As shown in these two figures, the available renewable energy shows a diurnal pattern. This is due to the fact that the local weather conditions have a nearly diurnal pattern.

4.3.3 Real-World Workload Traces

To build our workloads in the simulator, we use a trace of Internet traffic from Wikipedia.org [64]. In particular, we use this tracefile with 2-month long data, which contains 10% of user requests that arrived at Wikipedia between October 1st, 2007 and November 30th, 2007. Figure 4.4 shows the hourly behavior of user requests in October and November, 2007. As illustrated in the figure, the users' behavior shows a very clear weekly pattern in visiting the Wikipedia website. Specifically, 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 [65], can also be integrated into our system.

To make our evaluation more general, we also stress test GreenWare with another workload trace from the 1998 World Cup game, which includes the request data of 33 servers from 4 geographical locations. In particular, it records the incoming requests to all the servers with a granularity of 1 second from April 30th to July 26th, 1998.

4.3.4 Electricity Price Traces

To simulate the electricity price for the brown energy, we use the price trace from New York Independent System Operator (NYISO) [13], since they have complete and accurate price data records. Specifically, we use the Day-Ahead price data from November 1st, 2007 to November 30th, 2007, which is consistent with the dates of the Wikipedia traces. We apply the price data from the four zones, including Capital, Central, Dunwoodie and Genesee to the 4 data centers in our simulation.

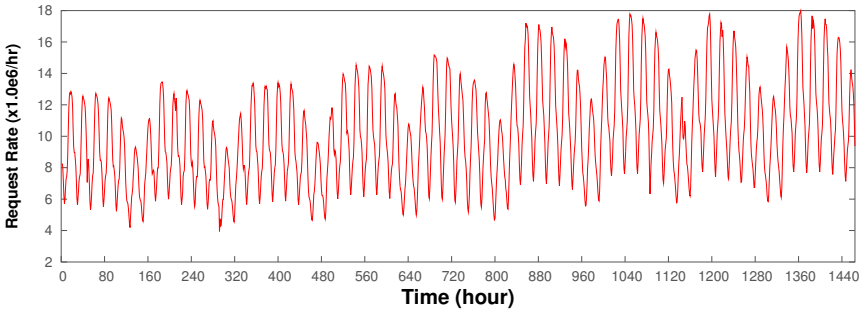

Figure 4.4: Wikipeida workload trace from October 1st, 2007 to November 30th, 2007.

On the other hand, regarding the electricity price of renewable energy, it is usually true that renewable energy has a higher electricity price compared to brown energy [2, 7], due to the intermittent nature of renewable energy sources such as wind and sunlight, as well as expensive facility investments and management. For example, renewable energy costs an additional 1.5 cents per KWh compared to the regular energy in the power market of Virginia [2]. Furthermore, solar energy is typically much more expensive than wind energy, due to the relatively high capital expenses [3, 15]. Thus, to be more practical, in our simulation we assume that the wind electricity price is 1.5 cents higher per KWh than brown energy [2]; while solar energy is 18.0 cents higher per KWh [3].

4.4 Evaluation Results

In this section, we first introduce two baselines. We then compare the proposed GreenWare middleware system against the baselines.

4.4.1 Baselines

In our work, we use two baselines in our experiments, a cost minimization only policy and a green energy usage maximization only policy, referred to as *Min-Cost* and *Max-Green*, respectively. (1) **Min-Cost.** Similar to GreenWare, Min-Cost also

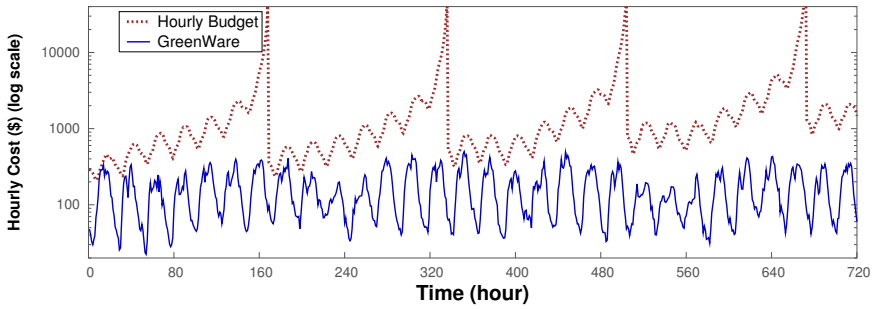

Figure 4.5: Hourly electricity cost by GreenWare with a sufficient monthly cost budget of $340K, with respect to Nov. 2007 Wikipedia trace.

tries to minimize the electricity cost by distributing requests among geographically distributed data centers to leverage the varying electricity prices in different locations. However, different from GreenWare, Min-Cost is unaware of renewable energy and thus prefers brown energy in cost minimization. Min-Cost is similar to the state-of-the-art work [58] in minimizing the electricity bill in operating data center networks. (2) **Max-Green.** Similar to GreenWare, Max-Green tries to maximize the use of renewable energy by distributing more requests to data centers where more renewable energy is available. However, Max-Green does so regardless of the cost budget and thus may lead to a high operation cost for the Internet service operators and sometimes even budget violations. This scheme is similar to the state-of-the-art work [61] in powering data centers with renewable energy.

4.4.2 Impacts of the Monthly Cost Budget

In this experiment, we evaluate the proposed GreenWare middleware with respect to different monthly cost budgets.

Figures 4.5 and 4.6 depict how GreenWare works with the Wikipedia workload under a monthly cost budget of $340K. In particular, these two figures show that with a sufficient monthly cost budget (*e.g.*, as shown in Figure 4.5, the allocated hourly budget is sufficient throughout the entire month), brown energy is used only in the invocation periods with insufficiently available renewable energy. That is, as indicated

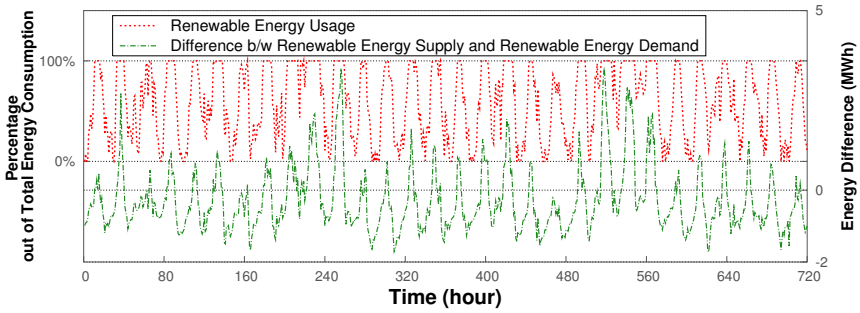

Figure 4.6: Hourly renewable energy usage by GreenWare with a sufficient monthly cost budget of $340K, with respect to Nov. 2007 Wikipedia trace.

in Figure 4.6, only when the available renewable energy supply is less than the actual renewable energy demand (*i.e.*, a difference lower than 0), the corresponding renewable energy usage does not reach 100%, *e.g.*, the hours of 2, 5, 6, 7 and etc. Note that there are some invocation periods which have a zero usage of renewable energy, *e.g.*, the hours of 1, 3, 4 and etc. This is because that there is no available renewable energy at all due to the weather conditions in those invocation periods. In addition, Figure 4.5 demonstrates that the hourly allocated cost budget within one week shows a growing trend. 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.

We then study GreenWare under a series of different monthly cost budgets. As shown in Figure 4.7, with the increase of the monthly cost budget, the monthly average percentage of renewable energy usage keeps rising and then stays stable. This is due to the fact that fewer invocation periods are allocated with an insufficient cost budget in the case with a higher monthly cost budget. Therefore, more renewable energy can be used to power the data center networks. For example, with a monthly cost budget of $100K, there are 202 invocation periods which have sufficient renewable energy supply but with an insufficient allocated cost budget; while as low as only 42 invocation periods are allocated with an insufficient cost budget in the case with a $160K monthly cost budget. As a result, a higher monthly average percentage

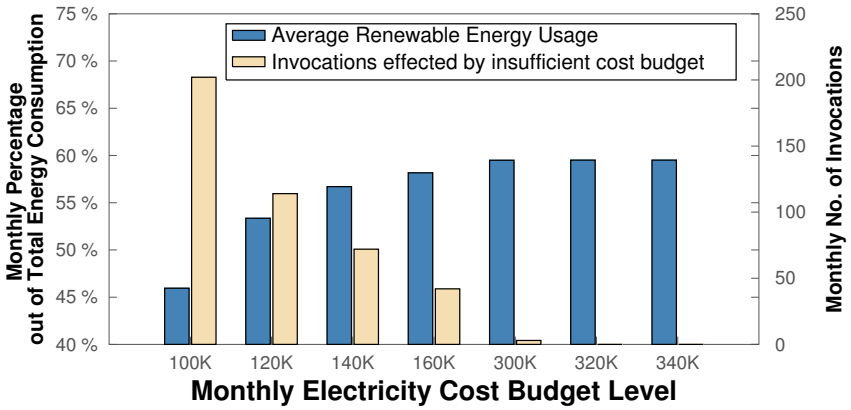

Figure 4.7: Average percentage of renewable energy usage by GreenWare with a series of different monthly cost budgets.

of 58.17% of renewable energy usage is achieved with the monthly cost budget of \$160K, compared to a percentage of 45.95% with the monthly budget \$100K. Thus, when all the invocation periods have a sufficient budget due to a sufficient monthly cost budget, *e.g.*, \$320K and \$340K, the monthly average renewable energy usage stays stable. This set of experiments demonstrates that GreenWare can significantly increase the use of renewable energy in powering the data center network, subject to the desired cost budget.

4.4.3 Comparison with Baselines

In this experiment, we compare GreenWare with the two baselines: Min-Cost and Max-Green.

Figure 4.8 depicts the cost and brown energy consumption of GreenWare, Max-Green and Min-Cost, with respect to a given monthly budget, *e.g.*, \$100K, for the Wikipeida workload. The results are normalized against Min-Cost, which actually indicates the case of only using brown energy in powering data center networks. Figure 4.8 shows that although Max-Green (*i.e.*, maximizing the use of green energy regardless of cost budget) can decrease brown energy consumption by 58% compared to Min-Cost by utilizing as much renewable energy as possible. However, due to

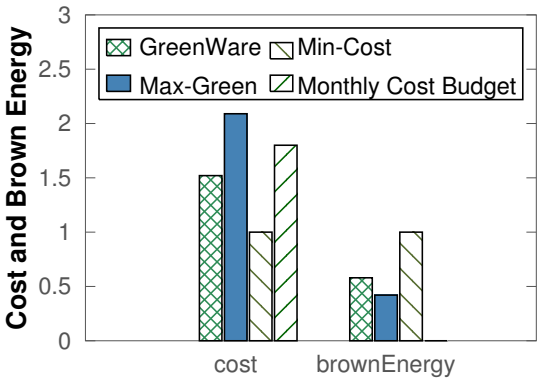

Figure 4.8: Comparison between GreenWare and baselines with respect to Nov. 2007 Wikipeida trace.

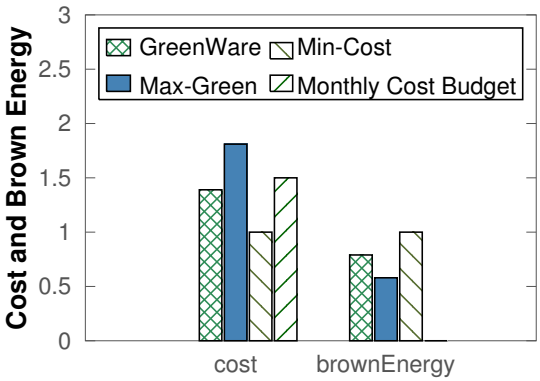

Figure 4.9: Comparison between GreenWare and baselines with respect to Jun. 1998 World Cup trace.

its unawareness of cost budget, Max-Green results in a 109% cost increase and exceeds the monthly cost budget by 29%. On the other hand, GreenWare can achieve an as-much-as-42% decrease in brown energy consumption at only a 52% cost increase, compared to Min-Cost. More importantly, GreenWare successfully controls the electricity bill to stay within the cost budget for the Internet service operator.

To demonstrate the effectiveness of GreenWare with different workloads, we also stress test GreenWare using the 1998 World Cup trace. Specifically, we use the request trace in June as the incoming workload in the simulation, and the May trace as historical data to predict the hourly cost budget. To simulate the workload of cloud-service data centers, we proportionally increase the request numbers. Figure 4.9 shows the experiment results on the comparison between GreenWare and the two baselines. As demonstrated in the figure, Max-Green achieves a 42% decrease in brown energy consumption compared to Min-Cost. However, the electricity bill exceeds the given monthly cost budget (*e.g.*, \$100K) by 31%. On the other hand, GreenWare obtains an as-much-as-21% decrease in brown energy consumption while successfully controlling the electricity bill to stay within the monthly cost budget.

4.4.4 Impacts of Pricing Policies of Renewable Energy

In this experiment, we show that the proposed GreenWare middleware always prefers the type of renewable energy that has a lower electricity price. Thus, an efficient cost minimization is guaranteed. Since in our work we just consider two types of the most popular renewable energy, *i.e.*, wind energy and solar energy, we assume two different pricing policies: (1) wind energy has a lower electricity price, as discussed in Section 4.3.4; and (2) solar energy has a lower price than wind energy. Note that the current practice is that wind energy is typically less expensive than solar energy. However, in order to stress test GreenWare, we assume a lower price for solar energy in (2).

Figures 4.10 and 4.11 demonstrate how the usage of different types of renewable energy varies with different pricing policies as discussed above. Intuitively, the more expensive renewable energy is taken into use only when the less expensive type of renewable energy is used up. As shown in Figure 4.10, with the first pricing policy (*i.e.*, wind energy price is lower), solar energy is used to power data centers only after all the supplied wind energy has been used up, as indicated in the second data center (DC#2). Similarly, with the second pricing policy, wind energy is used to power data centers only after all the available less expensive solar energy is consumed, as in all the data centers in Figure 4.11. Note that in Figure 4.10, Data Centers 1, 3 and 4 begin to use the more expensive solar energy though there is still some wind energy left. This is because that there are some invocation periods when the available wind energy is too much to serve the incoming workload. As a result, some wind energy is left unused and the unused wind energy cannot be used in the following invocation periods due to the intermittent feature of the renewable energy.

4.5 Summary

Two key questions faced by many cloud-service operators are 1) how to dynamically distribute service requests among data centers in different geographical locations,

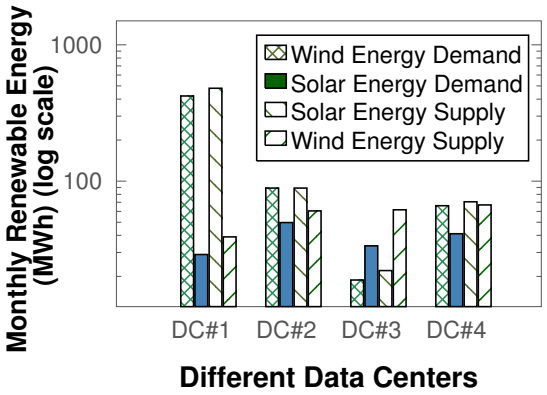

Figure 4.10: Monthly renewable energy usage by GreenWare when wind energy price is lower than solar energy price.

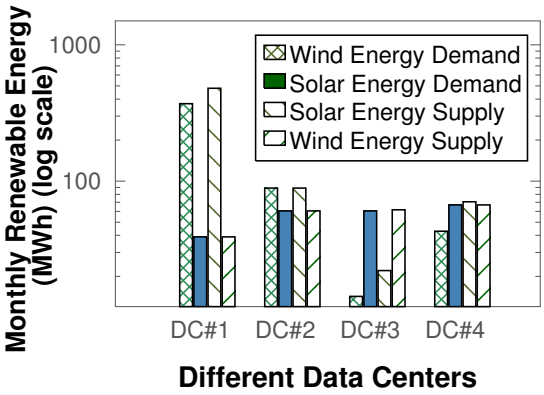

Figure 4.11: Monthly renewable energy usage by GreenWare when solar energy price is lower than wind energy price.

based on the local weather conditions, to maximize the use of renewable energy, and 2) how to do so within their allowed operation budgets. In this thesis, we have presented GreenWare, a novel middleware system that conducts dynamic request dispatching to maximize the percentage of renewable energy used to power a network of distributed data centers, subject to the desired cost budget of the Internet service operators. Our solution first explicitly models the intermittent generation of renewable energy, e.g., wind power and solar power, with respect to varying weather conditions in the geographical location of each data center. We then formulate the core objective of GreenWare as a constrained optimization problem and propose an efficient request dispatching algorithm based on linear-fractional programming (LFP). We evaluate GreenWare with real-world weather, electricity price, and workload traces. Our experimental results show that GreenWare can significantly increase the use of renewable energy in cloud-scale data centers without violating the desired cost budget, despite the intermittent supplies of renewable energy in different locations and time-varying electricity prices and workloads.

Chapter 5

Conclusion

With the rapid expansion on the number of hosted servers, high energy consumption has become one of the most serious concerns for large-scale data centers that are operated by Internet service providers. Therefore, 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, which can thus impact the power markets. In addition to high electricity bills, the enormous energy consumption of cloud-scale data centers can also lead to negative environmental implications (*e.g.*, CO_2 emission and global warming), due to their large carbon footprints. To reduce the negative environmental implications caused by the rapidly increasing energy consumption, many Internet service operators have started taking various initiatives to operate their cloud-scale data centers with renewable energy. Unfortunately, due to the intermittent nature of renewable energy sources such as

wind turbines and solar panels, currently renewable energy is often more expensive than brown energy that is produced with conventional fossil-based fuel. As a result, utilizing renewable energy may impose a considerable pressure on the sometimes stringent operation budgets of Internet service operators. In this thesis, solutions are discussed to reduce the electricity bill, as well as to green cloud-scale data centers for Internet service providers.

First, 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 that impact the power markets. 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.

Second, we propose GreenWare, a novel middleware system that conducts dynamic request dispatching to maximize the percentage of renewable energy used to power a network of distributed data centers, subject to the desired cost budget of the Internet service operator. Our solution first explicitly models the intermittent generation of renewable energy, e.g., wind power and solar power, with respect to varying weather conditions in the geographical location of each data center. We then formulate the core objective of GreenWare as a constrained optimization problem and propose an efficient request dispatching algorithm based on linear-fractional programming (LFP). Our experimental results show that GreenWare can significantly increase the use of renewable energy in cloud-scale data centers without violating the desired cost budget,

despite the intermittent supplies of renewable energy in different locations and time-varying electricity prices and workloads.

Bibliography

Bibliography

[1] BP MSX 120 solar module. http://pdf.directindustry.com/pdf/bp-solar/bp-msx-120-solar-module/15873-68158.html. 44, 49

[2] Dominion Virginia Power. http://www.dom.com/. 5, 6, 36, 51

[3] Energy modality comparison based on projected cents per kilowatt-hour. http://peswiki.com/. 51

[4] Green House Data: Greening the data center. http://www.greenhousedata.com/. 5

[5] Introduction to lp_solve 5.5.2.0. http://lpsolve.sourceforge.net/5.5/. 22

[6] ISO New England. http://www.iso-ne.com/. 13

[7] Los Angeles Department of Water & Power. http://www.ladwp.com/. 6, 51

[8] Measurement and Instrumentation data center. http://www.nrel.gov/midc/. 39, 48

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

[10] The Running Time of the Simplex Method. www.mpi-inf.mpg.de. 47

[11] Solar Electricity Prices. http://solarbuzz.com/. 6

[12] WindPower Program. http://www.wind-power-program.com/index.htm. 7, 37, 43

[13] NYISO. http://www.nyiso.com/, 1999. 50

[14] Data Center Knowledge. http://www.datacenterknowledge.com/, 2008. 22

[15] Solar Power at Data Center Scale. http://www.datacenterknowledge.com/, 2009. 5, 6, 36, 51

[16] Google Buys 20 Years' Worth of Wind Energy To Power Data centers. http://www.huffingtonpost.com/, 2010. 6

[17] F. Ahmad et al. Joint optimization of idle and cooling power in data centers while maintaining response time. In *ASPLOS*, 2010. 1, 3, 15, 19, 20, 22, 25, 38, 41

[18] M. Al-Fares et al. A scalable, commodity data center network architecture. In *SIGCOMM*, 2008. 19, 21

[19] A. O. Allen. *Probability, Statistics, and Queuing Theory with Computer Science Applications*. 1990. 19

[20] L. A. Barroso et al. The case for energy-proportional computing. *IEEE Computer*, pages 33–37, December 2007. 3

[21] G. Bolch et al. *Probability, Statistics, and Queuing Theory with Computer Science Applications*. 1998. 19

[22] G. Bolch et al. *Queueing Networks and Markov Chains*. Wiley-Interscience, 2005. 19, 41

[23] M. Brown and J. Renau. Rerack: Power simulation for data centers with renewable energy generation. In *GreenMetrics*, 2011. 11

[24] L. Castaner and S. Silvestre. *Modelling Photovoltaic Systems Using PSpice*. John Wiley & Sons, 2002. 44

[25] J. Chase et al. Managing energy and server resources in hosting centers. In *SOSP*, 2001. 1, 9

[26] G. Chen et al. Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI*, 2008. 1, 9

[27] 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*, 14, 2002. 17

[28] J. Ding and R. Radhakrishnan. A new method to determine the optimum load of a real solar cell using the lambert w-function. *Solar Energy Materials and Solar Cell*, 2008. 45

[29] E. Elnozahy et al. Energy-efficient server clusters. *Power-Aware Computer Systems*, pages 179–197, 2003. 1, 9

[30] X. Fan et al. Power provisioning for a warehouse-sized computer. In *ISCA*, 2007. 4

[31] G.N.Kariniotakis, G.S.Stavrakakis, and E.F.Nogaret. Wind power forecasting using advanced neural networks models. *IEEE transactions on Energy conversion*, pages 762–767, 1996. 43

[32] Goiri et al. Intelligent placement of datacenters for internet services. In *ICDCS*, 2011. 10

[33] S. Govindan et al. Benefits and limitations of tapping into stored energy for datacenters. In *ISCA*, 2011. 10

[34] A. Greenberg et al. The cost of a cloud: research problems in data center networks. *ACM SIGCOMM Computer Communication Review*, 39(1):68–73, 2008. 1, 3, 4, 6, 14

[35] B. Heller et al. Elastictree: Saving energy in data center networks. In *NSDI*, 2010. 3, 19, 21, 26

[36] J. Heo et al. Integrating adaptive components: An emerging challenge in performance-adaptive systems and a server farm case-study. In *RTSS*, 2007. 9

[37] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, 2005. 47

[38] A. Hohl. Wind Power for Data Centers. http://www.renewableenergyworld.com/rea/blog/post/2009/08/wind-power-for-data-centers, 2009. 43

[39] T. Horvath et al. Dynamic voltage scaling in multitier web servers with end-to-end delay control. *IEEE Transactions on Computers*, pages 444–458, 2007. 9

[40] L. Kothari, P. Mathur, A. Kapoor, P. Saxena, and R. Sharma. Determination of optimum load for a solar cell. *Journal of Applied Physics*, pages 5982–5984, 2009. 45

[41] K. Le, R. Bianchini, M. Martonosi, and T. D. Nguyen. Cost- and energy-aware load distribution across data centers. In *HOTPOWER*, 2009. 10

[42] K. Le, O. Bilgir, R. Bianchini, M. Martonosi, and T. D. Nguyen. Managing the cost, energy consumption, and carbon footprint of internet services. In *SIGMETRICS*, 2010. 1, 2, 10, 38

[43] K. Le et al. Reducing electricity cost through virtual machine placement in high performance computing clouds. In *SC*, 2011. 10

[44] C. Li, A. Qouneh, and T. Li. Characterizing and analyzing renewable energy driven data centers. In *SIGMETRICS*, 2011. 11

[45] C. Li, W. Zhang, C. B. Cho, and T. Li. Solarcore: Solar energy driven multi-core architecture power management. In *HPCA*, 2011. 7, 37, 44, 45, 49

[46] F. Li. Continuous locational marginal pricing (CLMP). *IEEE Transactions on Power Systems*, 22(4):1638 –1646, 2007. 13, 14, 15, 22

[47] F. Li et al. Congestion and price prediction under load variation. *IEEE Transactions on Power Systems*, 24(2):911–922, 2009. 3, 13, 14, 22, 23

[48] J. Li et al. Towards optimal electric demand management for internet data centers. In *Techreport*, 2010. 19, 21, 26, 27, 48

[49] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. In *INFOCOM*, 2011. 10, 48

[50] R. Miller. Who Has the Most Web Servers? http://www.datacenterknowledge.com/archives/2009/05/14/whos-got-the-most-web-servers/, 2009. 3, 14, 25

[51] 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. 19

[52] M. R. Patel. *Power systems: Design, Analysis, and Operation*. CRC Press, 2006. 7, 37, 43

[53] M. V. Paukshto and K. Lovetskiy. Invariance of single diode equation and its application. In *PVSC*, 2008. 44

[54] T. Petru and T. Thiringer. Modeling of wind turbines for power system studies. *IEEE transactions on Power Systems*, pages 1132–1139, 2002. 43

[55] G. Pistoia. *Battery Operated Devices and Systems: From Portable Electronics to Industrial Products*. Elsevier, 2011. 5

[56] A. Qureshi et al. Cutting the electric bill for internet-scale systems. In *SIGCOMM*, 2009. 1, 2, 3, 6, 10, 14, 17, 34, 38, 39

[57] R. Raghavendra et al. No power struggles: Coordinated multi-level power management for the data center. In *ASPLOS*, 2008. 4

[58] L. Rao et al. Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM*, 2010. 2, 10, 14, 20, 25, 26, 27, 34, 38, 41, 42, 48, 52

[59] D. Sera, R. Teodorescu, and P. Rodriguez. PV panel model based on datasheet values. In *ISIE*, 2007. 7, 37, 44, 45

[60] A. Services. Peak Power Rebate Overview. http://www.ameren.com. 3

[61] C. Stewart and K. Shen. Some joules are more precious than others: Managing renewable energy in the datacenter. In *HOTPOWER*, 2009. 11, 52

[62] P. Thibodeau. Wind power data center project planned in urban area. http://www.computerworld.com/, 2008. 5

[63] G. Trecate et al. Optimization with piecewise-affine cost functions. Technical report, Technical Report AUT01-13, 2001. 22

[64] G. Urdaneta et al. Wikipedia workload analysis for decentralized hosting. *Elsevier Computer Networks*, 53(11):1830–1845, July 2009. http://www.globule.org/publi/WWADH_comnet2009.html. 26, 50

[65] B. Urgaonkar et al. Dynamic provisioning of multi-tier internet applications. In *ICAC*, 2005. 27, 50

[66] R. Urgaonkar et al. Optimal power cost management using stored energy in data centers. In *SIGMETRICS*, 2011. 10, 38

[67] A. Verma et al. Server workload analysis for power minimization using consolidation. In *USENIX ATC*. USENIX Association, 2009. 1, 9

[68] X. Wang et al. Cluster-level feedback power control for performance optimization. In *HPCA*, 2008. 4

[69] Y. Zhang, Y. Wang, and X. Wang. Capping the electricity cost of cloud-scale data centers with impacts on power markets. In *HPDC*, 2011. 10, 38

Vita

Yanwei Zhang was born in Weihai, Shandong Province, China. She received her B.S. degree in Computer Science, as well as in Economics in the year of 2008 from Shandong University, China. Before started her graduate study in University of Tennessee, Knoxville, she worked as a research assistant at Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. She officially joined in University of Tennessee, Knoxville in January of the year 2010.