2013 IEEE 27th International Symposium on Parallel & Distributed Processing

Profit Aware Load Balancing for Distributed Cloud Data Centers

Shuo Liu*, Shaolei Ren†, Gang Quan*, Ming Zhao†, and Shangping Ren‡
*Department of Electrical and Computer Engineering, Florida International University, Miami, FL, 33174
†School of Computing and Information Sciences, Florida International University, Miami, FL, 33199
‡Department of Computer Science, Illinois Institute of Technology, Chicago, IL, 60616
Emails: {sliu005, gang.quan}@fiu.edu, {sren, ming}@cs.fiu.edu, ren@iit.edu

***Abstract*—The advent of cloud systems has spurred the emergence of an impressive assortment of Internet services. Recent pressures on enhancing the profitability by curtailing surging dollar costs on energy have posed challenges to, as well as placed a new emphasis on, designing energy-efficient request dispatching and resource management algorithms. What further adds to the design challenge is the highly diverse nature of Internet service requests in terms of Quality-of-Service (QoS) constraints and business values. Nonetheless, most of the existing job scheduling and resource management solutions are for a single type of request and are profit oblivious. They are unable to reap the benefit of multi-service profit-aware algorithm designs.**

In this paper, we consider a cloud service provider operating geographically distributed data centers in a multi-electricity-market environment, and propose an energy-efficient, profit- and cost-aware request dispatching and resource allocation algorithm to maximize a service provider's net profit. We formulate the net profit maximization issue as a constrained optimization problem, using a unified task model capturing multiple cloud layers (e.g., SaaS, PaaS, IaaS.) The proposed approach maximizes a service provider's net profit by judiciously distributing service requests to data centers, powering on/off an appropriate number of servers, and allocating server resources to dispatched requests. We conduct extensive experiments to validate our proposed algorithm. Results show that our proposed approach can improve a service provider's net profit significantly.

I. Introduction

With the development of cloud computing, service providers are able to provide a variety of complex applications and services to people's daily lives, such as Google Docs and AppEngine, Amazon EC2 and S3, etc. These applications and services are all supported by service provider's data centers and delivered to a wide range of clients over the Internet.

The large number of service requests drastically increases not only the need for data centers, but also the scale of data centers and their energy consumptions. The dollar cost spent on energy consumption takes a large portion of a service provider's operational cost annually. As an example, Google has more than 500K servers and it consumes more than $38M worth of electricity each year. Similarly, Microsoft has more than 200K servers and spends more than $36M on electricity annually [1]. Evidently, dollar costs on energy consumptions have been a critical part in operational cost for service providers. It is fair to say that an efficient computing resource management approach for distributed cloud data centers is essential to service providers.

A well-designed resource management scheme can effectively reduce the dollar cost on energy consumptions. This is particularly true for distributed cloud data centers where their dollar costs on energy are sensitive to factors such as workload distribution, data transferring, electricity prices, etc. The problem, however, is how to take all of these factors into consideration when designing and developing a resource management scheme such that the QoS with respect to different service requests can be satisfied, as well as its cost on energy consumptions can be minimized.

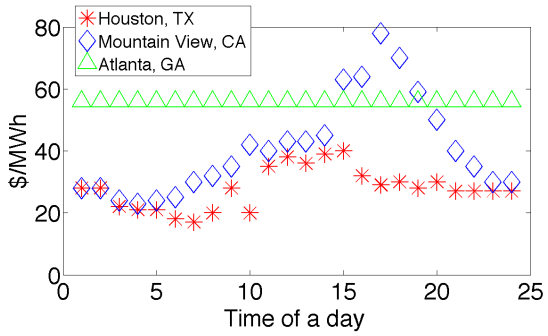


Figure 1. Electricity prices at different locations in a day.

In this paper, we present a profit- and cost-aware resource management approach for distributed cloud data centers to optimize a service provider's net profit (defined as the profit minus the dollar cost on energy.) Service providers gain profit by satisfying service requests to the level identified based on a certain service level agreement (SLA). At the same time, service providers need to pay the cost for energy consumed by transferring and processing requests. For several reasons, e.g. high availability, disaster tolerance, and uniform response times, etc., service providers usually spread their data centers in a wide geographical region. The electricity prices at different data center locations vary differently throughout a day. Therefore, opportunities present to reduce the dollar cost on electricity by selecting proper data centers for processing service requests. By taking advantages of the multi-electricity-market (as shown in Figure 1 [2]),

1530-2075/13 $26.00 © 2013 IEEE
DOI 10.1109/IPDPS.2013.60

our approach has a high efficiency of energy and computing resource usage by judiciously dispatching service requests to different data centers, powering on an appropriate number of servers at different data centers, and adaptively allocating resources to these service requests. Multiple types of services, with no priority difference, are considered in our model. Even though there are various layers in cloud computing, such as SaaS, PaaS, and IaaS, we do not focus on any special layer. Instead, we abstract the service requests of those layers with a uniform task model. Compared with related work, our contributions in this paper can be summarized as follows:

- We propose a system model that incorporates the multi-electricity-market, SLA, and net profit into a single unified resource management framework. To our best knowledge, this is the first work that deals with multi-electricity-markets, multiple types of requests, and multi-level SLAs, simultaneously.
- We model the profit gained by a service provider as a multi-level step-downward function, which is capable of simulating various scenarios (as explained in Section III-B1.) We formulate our problem of determining how to dispatch service requests to different data centers, how many servers should be powered on in each data center, and how computing resources should be allocated to service requests as a constrained optimization problem. We also derive a series of constraints to simplify the implementation of our approach.
- The effectiveness of our proposed approach is validated through simulations on both synthetic workload and real data center traces with true electricity price history.

The remainder of the paper is organized as follows. Section II introduces the background of our problem and related work. Our system architecture and task model are proposed in Section III. Section IV discusses our approach in detail. Experimental results are presented in Sections V, VI and VII. We conclude in Section VIII.

II. Background and related work

Task scheduling and resource management are critical to ensure the QoS (defined by SLA), and energy saving or energy dollar cost reduction. There has been extensive research work conducted for optimizing a data center's energy consumption or cutting down the electricity bills for service providers. This work can be largely divided into two groups. One is SLA-based resource management for a single data center and the other is for distributed data centers in a multi-electricity-market environment.

A. Single data center

Many types of SLA-based resource management research were conducted to lower energy consumptions or cut down the operational costs spent on energy consumptions. In [3], Chase et al. presented an architecture for resource management in a hosting center operating system. They adaptively provisioned server resources according to the offered workload. The efficiency of server clusters was improved by dynamically resizing the active server set in accordance with SLAs to respond to power supply disruptions or thermal events. Wang et al. in [4] solved a problem of managing power consumption in multi-tier web clusters equipped with heterogeneous servers. Their method employs dynamic voltage scaling (DVS). By adjusting the number of powered-on servers and their working frequencies, they effectively reduced the energy consumption in their web clusters. Different from our work, these two studies are focused on a single data center rather than distributed data centers. In addition, they only consider a single type of service request.

Liu et al. in [5][6] studied a method for multi-tier architecture that decides the workload distribution and computing capacity allocation to optimize the SLA-based profit a data center may achieve. However, this work does not account for the energy consumed by data centers. Later, in [7], energy consumption was considered and an energy consumption control method was proposed to satisfy certain SLAs and energy constraints. Contrary to the work in [5][6][7], our approach is for distributed cloud data centers in a multi-electricity-market environment.

Lin et al. [8] analytically formulated their optimal offline solution and developed the corresponding online algorithm to bound the number of powered-on servers with respect to certain delay constraints, in order to reduce the energy consumption for power-proportional data centers. Their approach focuses on a single service type, and implies that once the number of powered-on servers is fixed, the optimal dispatching rule will evenly distribute workloads across the servers. However, this is not suitable for multiple types of requests.

Recently, Sarood et al. [9] proposed models that take cooling energy consumptions into consideration. By reasonably balancing workloads and employing dynamic voltage and frequency scaling (DVFS), they successfully lowered the overall energy consumed by their cooling system while satisfying temperature constraints. Cooling factors are out of the scope of our work. However, our model can be extended by adding a parameter describing a data center's power utilization efficiency (PUE) to account for the energy consumed by cooling systems as well as other peripheral equipments.

B. Distributed data centers

Le et al. have studied the advantages of using green energy (e.g. energy generated by winds or solar energy). These studies help to replace the usage of "brown" energy (produced via carbon-intensive means) with "green energy" during a data center's operation in order to cut down the

cost spent on energy consumptions. For instance, a study of a framework for multi-data-center services was introduced in [10][11]. However, no SLA-based profit was considered in these studies. Only response time constraints were considered to reflect the QoS requirements.

Since most cloud systems geographically distribute their data centers, requests dispatching and resource management design for multiple data centers attracts more and more attention. The research in [2][12] extended the work in [4] to a distributed data center architecture in a multi-electricity-market environment. Rao et al. modeled their problem as a constrained mixed-integer linear programming, and proposed an efficient approach to approximate the problem with a linear programming formulation. These studies only considered a single service type. Our new proposed algorithm works for multiple types of service requests. Moreover, our model accounts for transferring costs as well.

In real-time services, QoS is reflected by a service's timeliness. After Jensen first proposed the time utility function (TUF) [13], there were many studies conducted based on TUFs to study the timeliness of real-time tasks in various fields [14][15][16]. Most of them are task-level scheduling algorithms. Scheduling activities are performed according to each single task's behavior. In [17], Liu et al. proposed a task allocation and scheduling algorithm for distributed data centers in a multi-electricity-market environment. They implemented two TUFs to describe each task's potential profit and penalty, respectively. The scheduling algorithm accounts for the dollar costs of data transferring and processing. Nevertheless, the work in [17] has a high timing complexity for online implementation in network-based system because of the huge amount of service requests. Our new proposed approach is a significant improvement of [17] by using the queuing theory to build a constrained optimization formula in order to flexibly dispatch requests and allocate computing resources for maximizing net profits in distributed cloud data centers. The system models, approaches and techniques are all fundamentally different from [17]. Instead of focusing on each single service request [17], our new approach focuses on each type of requests. Requests of the same service type follow the same scheduling policy.

III. System models and problem formulation

In this section, we introduce our system model, based on which we develop our time-slotted profit-aware request dispatching and resource management approach for distributed cloud data centers in a multi-electricity-market environment. Our approach periodically runs at the beginning of each time slot T based on the average arrival rates during a slot since job interarrival times are much shorter compared to a slot [8]. Requests arrival pattern forecast is not studied in our work. Existing prediction methods (e.g. the Kalman Filter [18],) or studies (e.g. [19][20]) that have been conducted can be employed if necessary. The length of T is a pre-defined constant that is decided by several factors, e.g. adjusting frequencies of electricity prices (electricity prices stochastically vary over time due to the deregulation of electricity market [21].) We consider that the electricity prices in a time-slot T are constant. Constant prices during a time period are widely implemented in prior work [8][21].

A. System architecture

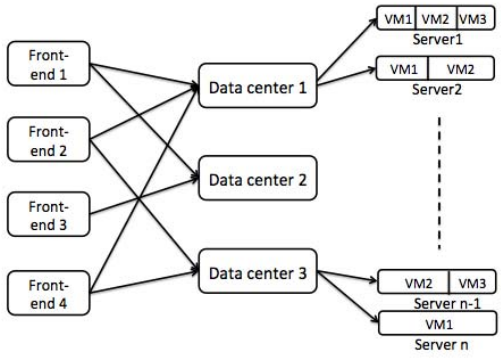


Figure 2. System architecture.

A typical distributed cloud data center system can be illustrated by Figure 2. In our system, service requests come from various places and are collected by S nearby front-end servers, where $S = s_1, s_2, ..., s_S$. Then, requests are dispatched to I capable servers in L data centers via network according to a related metric, where $I = I_1, I_2, ..., I_I$ and $L = l_1, l_2, ..., l_L$.

Virtualization technology, which boosts the realization of the long-held dream of computing as a utility [22], is employed in our architecture to enable server consolidation and simplify computing resource sharing in physical servers. Elasticity of virtualization helps improve the usage efficiencies of computer resources and energy. Different types of services can be held in the same server within their own virtual machines (VMs). The same CPU can be shared by different VMs when necessary. We assume that once a server is powered on, it always runs at its maximum speed. In our scenario, the data centers are heterogeneous, and the servers in a data center are homogeneous. It can be easily extended to heterogeneous data centers with heterogeneous servers.

B. Task model

Requests in our system are soft real-time in nature and may encounter both profit and cost. Profit comes from successfully guaranteeing the average delay satisfaction for each type of request [23]. Cost is the dollar cost spent on transferring and processing requests.

1) Profit: TUFs are able to precisely specify the semantics of soft real-time constraints [14]. It indicates that in real-time systems, when tasks are completed with respect to their time constraints, the system will be assigned values

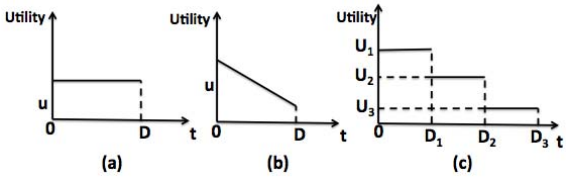


Figure 3. Typical TUFs.

that vary with the finishing times of the tasks. A TUF can be in any shape. Commonly used TUFs include a constant value before its deadline (Figure 3(a)), a monotonic non-increasing function (Figure 3(b)), or a multi-level step-downward function (Figure 3(c)), etc. In our scenario, all types of requests desire quick responses. It means that the earlier the tasks are finished, the more utilities they assign to their system. We employ TUFs to represent the profits of processing requests, which are non-increasing functions. Non-increasing TUFs match the SLAs well, since longer delays (beyond some defined time instances) result in lower profits. We will analyze constant value TUFs and multi-level step-downward TUFs in the following sections. These two types of TUFs are representative, especially the multi-level step-downward TUFs. A monotonic non-increasing TUF can be simulated by using a special multi-level step-downward TUF, which has an infinite number of steps. A constant TUF can be simulated as well by using step-downward TUF that only has one step. Consequently, a multi-level step-downward TUF is able to represent a wide range of scenarios and it explains why we mainly focus our study on multi-level step-downward TUFs.

Based on the queuing theory, i.e. M/M/1 queue (assuming that the request arriving follows poisson distribution,) it is not difficult to model the expected delay time for k-type requests as [24]

$$R_k = \frac{1}{\phi_k C \mu_k - \lambda_k} \tag{1}$$

where in Equation 1, C is a server's capacity, and is normalized to 1 in our scenario. In heterogeneous systems, different hardware configurations may have different capacities. μ_k is the processing rate for k-type requests with full capacity. Note that a server's resource may be shared by many different VMs at the same time. Therefore, its actual processing rate may not be μ_k. ϕ_k indicates the percentage of CPU resource allocated to k-type requests in a single server. λ_k is the arrival rate of k-type requests.

2) Cost: Cost consists of two parts. One is the dollar cost for processing requests, the other is the dollar cost for transferring requests.

Processing cost mainly comes from a server's energy consumption. The energy consumption in our work follows the model studied by Google [25] instead of traditional server's energy model. It is based on the energy consumption for processing each single service request. We believe this model is closer to the goal of converting computing ability into one kind of utility in people's daily lives (e.g. electricity.) Then, computing capacity usages are converted into utility consumptions. We assume that energy attributions of the requests are profiled, then the dollar cost on energy consumption for processing requests in a time slot can be expressed as follows:

$$PCost_k = P_k \times \lambda_k \times T \times p \tag{2}$$

where $PCost_k$ is the dollar cost for processing k-type requests. λ_k is k-type requests arriving rate. P_k is the energy attribution of k-type requests. Google's study shows that each web search costs $0.0003KWh$ on average. T and p are the length of a time slot (e.g. one hour, which is the same as the electricity prices' changing frequency) and the electricity price at the data center location in a time slot (as shown in Figure 1,) respectively.

Dollar cost on transferring requests from a front-end server to a corresponding data center is calculated in a similar method as the one in [1]. As shown in Equation 3, it is the product of unit transferring cost ($TranCost_k$) of each type of request, the distance between the request's origination and destination ($Distance_k$), arrival rate λ_k and the length of a time slot. Since requests may have various characteristics (e.g. sizes,) "$TranCost_k$" is employed to reflect the differences among requests.

$$TCost_k = TranCost_k \times Distance_k \times \lambda_k \times T \tag{3}$$

C. Problem formulation

With our system architecture and system model defined above, formally, our problem can be formulated as follows:

Problem 1: Given service requests and data center architectures as described above, develop an efficient on-line profit- and cost-aware workload dispatching and resource allocation approach to maximize net profits for service providers.

IV. Our approach

In this section, we introduce our approach in detail. For clarity, parameters used in this work are summarized in Table I. We formulate the solution for Problem 1 as a constrained optimization problem [26][27]. The results are used to decide request dispatching, resource allocation, and the number of servers that should be powered on.

The objective function of Problem 1 can be mathematically formulated as follows:

$$max \sum_{s=1}^{S} \sum_{l=1}^{L} \sum_{i=1}^{M} \sum_{k=1}^{K} \{U_k(R_{k,i,l})\lambda_{k,s,i,l} - Cost_{k,s,i,l}\lambda_{k,s,i,l}\}T \tag{4}$$

Parameters	Definations
K	number of service types in the system.
S	number of front-end servers in the system.
L	number of data centers in the system.
M_l	number of homogeneous servers in data center l.
$C_{i,l}$	capacity of server i in data center l.
μ_k	service rate for k-type requests at a server of capacity 1.
$\lambda_{k,s,i,l}$	k-type requests dispatched to server i in data center l comes from front-end server s.
$\phi_{k,i,l}$	CPU share for k-type requests at server i in data center l.
$R_{k,i,l}$	delay time for k-type requests at server i in data center l.
U_k	utility function for k-type requests. U_k^q corresponds to the utility in q_{th} level.
$D_{k,q}$	relative sub-deadline for the q_{th} utility level.
D_k	relative deadline for k-type requests.
$P_{k,l}$	energy cost for processing k-type requests in data center l.
p_l	electricity price at data center l at time t.
$d_{s,l}$	distance between front-end server f and data center l.
$PCost_{k,l}$	processing cost of k-type requests at data center l.
$TranCost_k$	unit transferring cost of k-type requests.

Table I
PARAMETER NOTATION.

After we substitute the factors in Equation 4 with Equation 1, 2 and 3, it becomes to Equation 5:

$$max \sum_{s=1}^{S}\sum_{l=1}^{L}\sum_{i=1}^{M}\sum_{k=1}^{K}\{U_k(R_{k,i,l})\lambda_{k,s,i,l} - P_{k,l}\lambda_{k,s,i,l}p_l - TranCost_k d_{s,l}\lambda_{k,s,i,l}\}T \quad (5)$$

with following constraints:

$$\frac{1}{\phi_{k,i,l}C_{i,l}\mu_{k,l} - \lambda_{k,s,i,l}} \leq D_k,\ \forall k,i,s,l \quad (6)$$

$$\sum_{s=1}^{S}\sum_{l=1}^{L}\sum_{i=1}^{M}\lambda_{k,s,i,l} \leq \sum_{s=1}^{S}\lambda_{k,s},\ \forall k \quad (7)$$

$$\sum_{k=1}^{K}\phi_{k,i,l} \leq 1,\ \forall i,s,l \quad (8)$$

Constraint 6 shows the QoS requirement. The average delay for each type of request cannot exceed its deadline. Constraint 7 assures that the number of assigned requests does not exceed the number of total service requests coming from the Internet. Constraint 8 bounds the CPU share by various types of services in a single server.

In our constrained optimization formulae, $\phi_{k,i,l}$ and $\lambda_{k,s,i,l}$ are the two variables that need to be solved, representing where to assign and how much workload should be assigned from each front-end server. In addition, as we know how requests are dispatched, we can determine how many servers should be powered on. Clearly, when there is no workload on a server, the server should be powered off.

In our model, we assume that server switching costs and durations are negligible compared to the total energy consumption and time of processing and transferring requests during a time slot (e.g. one hour.)

The complexity of our objective function depends heavily on the format of the utility function used to reflect a request's potential profit. Since multi-level step-downward TUFs are representative and cover a large scenario diversity, in what follows, we discuss three typical multi-level step-downward utility functions, and corresponding solutions for each of them. As stair TUFs need "*if-else*" descriptions, which are unfortunately not well supported by some popular non-linear mathematic programming (or some constraint logic programming) solvers, e.g. Prolog, we hence transform the "*if-else*" into a set of constraints.

1) One-level step-downward TUF: The first type of TUF has a constant utility before deadline and can be expressed as follows:

$$U_k = TUF(R_k) = \begin{cases} U_{k,1} & 0 < R_k \leq D_k \\ 0 & R_k > D_k \end{cases} \quad (9)$$

where, U_k is the utility of k-type requests. $U_{k,1}$ is a constant value. Before delay time R_k exceeds deadline D_k, U_k equals to $U_{k,1}$.

With one-level step-downward TUF, the objective function (Equation 5) is simply a linear function. Even though there is a nonlinear component in Equation 6, it can be linearized through simple transformations, i.e. $\phi_{k,i,l}C_{i,l}\mu_{k,l} - \lambda_{k,s,i,l} \geq \frac{1}{D_k}$. The whole problem can be solved by using traditional linear programming solvers [28].

2) Two-level step-downward TUF: This type of TUF can be expressed as follows:

$$U_k = TUF(R_k) = \begin{cases} U_{k,1} & 0 < R_k \leq D_{k,1} \\ U_{k,2} & D_{k,1} < R_k \leq D_k \\ 0 & R_k > D_k \end{cases} \quad (10)$$

where U_k is the utility of k-type requests. R_k is the delay time of k-type requests. $D_{k,q}$ is the relative sub-deadline for each utility level $U_{k,q}$, and q is the index of each level (i.e. the q-th sub-deadline of k-type requests to achieve the q-th utility level.) We assume that D_k is the final deadline for k-type service requests. Executing a request becomes meaningless once the delay time exceeds D_k.

Note that when the TUF employs a two-level step-downward function, the objective function is no longer a linear one. Furthermore, with Equation 10, it is challenging to formulate the objective in one formula. To solve this problem, we transform Equation 10 to a set of extra constraints as follows:

$$U_k \in \{U_{k,1}, U_{k,2}\}, (U_{k,1} > U_{k,2}) \tag{11}$$

$$(R_k - D_{k,1}) + \uplus(U_k - U_{k,1}) <= 0 \tag{12}$$

$$(D_{k,1} + \delta - R_k) + \uplus(U_{k,2} - U_k) <= 0 \tag{13}$$

where, $\uplus$ is a large constant. δ is a constant time value which is small enough. $D_{k,1} + \delta$ indicates the time instance that immediately follows time $D_{k,1}$.

To see the reason that Equation 10 can be equivalently transformed to a set of constraints listed in Equations 11, 12 and 13, consider the following two cases:

- When $0 < R_k \leq D_{k,1}$

  Under this condition, we readily have $R_k - D_{k,1} \leq 0$. From Equation 11, U_k can be either $U_{k,1}$ or $U_{k,2}$. Therefore, to satisfy Equation 13, we must have $U_k = U_{k,1}$. In the meantime, Equation 13 can be easily satisfied as long as $\uplus$ is large enough. To this end, $U_k = U_{k,1}$ is the only solution when $0 < R_k \leq D_{k,1}$.

- When $R_k > D_{k,1}$

  Under this condition, we readily have $D_{k,1} + \delta - R_k \leq 0$. Since U_k can be either $U_{k,1}$ or $U_{k,2}$, to satisfy Equation 12, we must have $U_k = U_{k,2}$. In the meantime, Equation 12 can be easily satisfied as long as $\uplus$ is large enough. To this end, $U_k = U_{k,2}$ is the only solution when $R_k > D_{k,1}$.

While we can transform Equation 10 to a set of constraints listed in Equations 11 – 13, the problem is not over. Note that Equation 11 is still a constraint that is not formulated properly. To formulate the constraint in Equation (11), we can define an integer variable x with

$$0 \leq x \leq 1 \tag{14}$$

such that

$$U = xU_{k,1} + (1 - x)U_{k,2} \tag{15}$$

With the extra constraints listed in Equations 11 - 13, it is desirable to use traditional integer linear programming solver to solve the problem. Unfortunately, this is not feasible. From Equation 1, it is not difficult to see that both Constraints 12 and 13 are non-linear formulae. To solve this problem, we need to employ the constraint logic programming solvers or nonlinear mathematic programming solvers such as ILOG CPLEX [29] and AIMMS [30] to find the near optimal solutions. With the help of the series of constraints, people may avoid the difficulty of implementing "$if, else$" statement in some solvers. Similar series can be derived for multi-level step-downward TUFs.

3) Three or more level step-downward TUF: This type of TUF can be formulated as follows:

$$U_k = TUF(R_k) = \begin{cases} U_{k,1} & 0 < R_k \leq D_{k,1} \\ U_{k,2} & D_{k,1} < R_k \leq D_{k,2} \\ U_{k,3} & D_{k,2} < R_k \leq D_{k,3} \\ \vdots & \\ 0 & R_k > D_k \end{cases} \tag{16}$$

Similarly, Equation 16 can be transformed into a series of new constraints as listed below:

$$U_k \in \{U_{k,1}, U_{k,2}, U_{k,3}, ..., U_{k,n}\}$$

$$\begin{aligned} (R_k - D_{k,1}) + \uplus(U_k - U_{k,1}) &<= 0 \\ (D_{k,1} + \delta - R_k) + \uplus(U_{k,2} - U_k)(U_k - U_{k,3}) &<= 0 \\ (R_k - D_{k,2}) + \uplus(U_{k,2} - U_k)(U_k - U_{k,1}) &<= 0 \\ (D_{k,2} + \delta - R_k) + \uplus(U_{k,3} - U_k)(U_k - U_{k,4}) &<= 0 \\ (R_k - D_{k,3}) + \uplus(U_{k,3} - U_k)(U_k - U_{k,2}) &<= 0 \\ &\vdots \\ (D_{k,n-1} + \delta - R_k) + \uplus(U_{k,n} - U_k) &<= 0 \end{aligned} \tag{17}$$

where, $U_{k,1} \ldots U_{k,n}$, $D_{k,1} \ldots D_{k,n}$, $\uplus$, and δ are the same as those in a two-level step-downward function, and $U_{k,1} > U_{k,2} > \cdots > U_{k,n}$. Take $n = 3$ as an example, we have:

$$U_k \in \{U_{k,1}, U_{k,2}, U_{k,3}\}, (U_{k,1} > U_{k,2} > U_{k,3}) \tag{18}$$

$$(R_k - D_{k,1}) + \uplus(U_k - U_{k,1}) <= 0 \tag{19}$$

$$(D_{k,1} + \delta - R_k) + \uplus(U_{k,2} - U_k)(U_k - U_{k,3}) <= 0 \tag{20}$$

$$(R_k - D_{k,2}) + \uplus(U_{k,2} - U_k)(U_k - U_{k,1}) <= 0 \tag{21}$$

$$(D_{k,2} + \delta - R_k) + \uplus(U_{k,3} - U_k) <= 0 \tag{22}$$

Equations 19 and 22 are very similar to Equations 12 and 13. The newly added constraints are Equations 20 and 21. Note that, similar to the analysis above, it is not difficult to see that as long as $\uplus$ is large enough, we must have $U_k = U_{k,1}$ when $R_k \leq D_{k,1}$, and $U_k = U_{k,3}$, when $R_k > D_{k,2}$ to satisfy Constraints 19 – 22.

Now, consider the situation when $D_{k,1} < R_k \leq D_{k,2}$ (note that $U_{k,1} > U_{k,2} > U_{k,3}$.) Under this condition, similarly, Equations 20 and 21 can be easily satisfied with any $U_k \in \{U_{k,1}, U_{k,2}, U_{k,3}\}$. From Equation 19, we can

conclude that, to satisfy Equation 21, we must have

$$(U_k - U_{k,1}) < 0$$

that is, we have either

$$U_k = U_{k,2} \quad or \quad U_k = U_{k,3} \tag{23}$$

Meanwhile, to satisfy Equation 22, we must have

$$(U_{k,3} - U_k) < 0$$

that is, we have either

$$U_k = U_{k,1} \quad or \quad U_k = U_{k,2} \tag{24}$$

Therefore, to satisfy Equations 19 - 22, we must have $U_k = U_{k,2}$ when $D_{k,1} < R_k \leq D_{k,2}$.

We have shown that Equation 16 can be transformed equivalently into Equation 17. The problem becomes how to formulate U_k (Equation 18) using a general form. Similarly, we can introduce an integer variable x, with

$$1 \leq x \leq n \tag{25}$$

where, n is the number of step levels. Then U_k can be formulated as follows:

$$U_k = \frac{\sum_{i=1}^{i \leq n} [\Pi_{j=0,j!=i}^{j \leq n}(j - x)] U_{k,i}}{(-1)^x x!(n-x)!} \tag{26}$$

As a result, U_k is successfully transformed into a series of constraints as described in Equations 17, 25, and 26. Same as above, together with our objective function, this constraint series can be solved by using constraint logic programming solvers or nonlinear programming solvers.

V. Study of basic characteristics

In this section, we study the basic characteristics of our new proposed approach using experiments with synthetic workloads and electricity prices. Experiments with real request traces and electricity prices will be shown in later sections.

A. Experiment setup

Two approaches were implemented and compared. One is called the "*Optimized*" approach, which is our new proposed one. The other one, called "*Balanced*," is a static approach that evenly dispatches workloads and allocates resources for every front-end server. During dispatching in the balanced approach, every front-end server first seeks the data center which has the lowest electricity price. Workloads are assigned to the servers in that data center first until its utilization is full. Then, workloads are forwarded to the rest data centers in accordance with the order of electricity prices, i.e. starting from the lowest data center to the highest one. Transferring cost is not considered in this basic study.

In this experiment, we set four front-end servers to collect and dispatch three types of service requests. There are three heterogeneous data centers with their own local electricity prices. Each data center has six homogeneous servers. For simplicity, the TUFs used here are those with constant values. Two groups of request arrival rates were set for simulating both light and heavy workloads (shown in Table II). Other test parameters for each server are summarized in Table III.

(a) Low arrival rates at every front-end server.

Front-end servers	request 1	request 2	request 3
$server1(\#/second)$	120	100	100
$server2(\#/second)$	150	100	50
$server3(\#/second)$	50	100	150
$server4(\#/second)$	50	50	100

(b) High arrival rates at every front-end server.

Front-end servers	request 1	request 2	request 3
$server1(\#/second)$	1120	500	1000
$server2(\#/second)$	1150	500	750
$server3(\#/second)$	550	1100	1150
$server4(\#/second)$	550	750	700

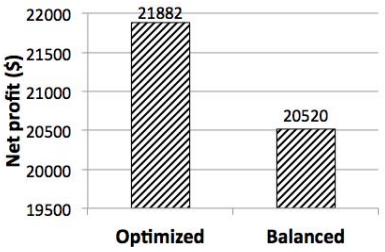
Table II
TWO REQUEST ARRIVAL SETS

Parameters	Datacenter1	Datacenter2	Datacenter3
C	1	1	1
μ_1, μ_2, μ_3 $(\#ofrequest/s)$	150, 130, 160	130, 120, 150	130, 130, 160
$cost_1$, $cost_2$, $cost_3$ (kWh)	2, 4, 6	1, 3, 5	1, 3, 6
p ($)	0.3	0.4	0.2

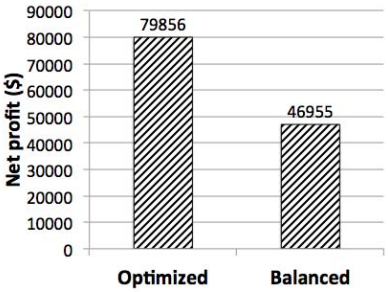
Table III
DATA CENTER PARAMETERS SETUP.

B. Experimental results

Figure 4(a) shows the comparison of net profits between "*Optimized*" and "*Balanced*" in a light workload. From the figure we can see that "*Optimized*" achieved a much higher net profit than "*Balanced*." Since "*Optimized*" takes overall factors into consideration, by balancing the trade-offs among various factors, e.g., profits, electricity prices, server capacities, etc., "*Optimized*" significantly outperforms "*balanced*" in terms of net profits. Similar for the heavy workload situation, as shown in Figure 4(b), "*Optimized*" obtains higher net profits. In addition, the result with heavy workload shows a higher efficiency of computing resource and energy utilization. Under the high arrival rate situation, even though none of the approaches was able to process all the requests, our proposed optimization-based approach processed around 16% more requests than the static method. Processing more requests leads to higher dollar costs on energy consumption, however, our proposed approach was able to cover the costs by obtaining more profits and resulted in a high net profit.



(a) Net profit with a low arrival rate.



(b) Net profit with a high arrival rate.

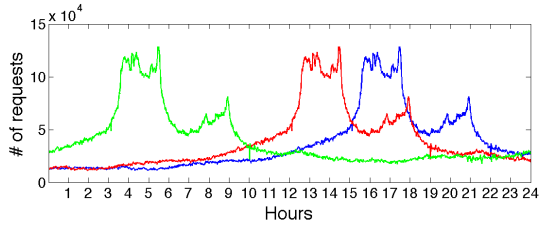
Figure 4. Experimental results with a low arrival rate.

VI. Study with real traces using one-level step-downward TUFs

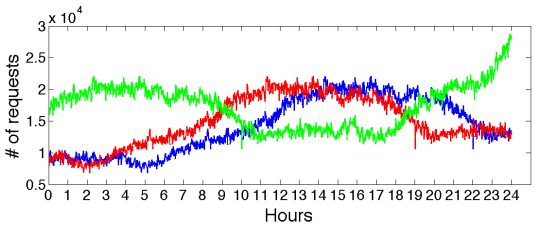
A. Experiment setup

In this study, we employed a real trace of the 1998 World Cup [31] to generate our service requests. We used a trace that contains requests spanning four different days as the service requests in a day collected by four front-end servers. There are three data center locations providing services to three types of requests dispatched from the four front-end servers. Each data center has six homogeneous servers. We simply shifted the request traces at a front-end server by some time units to simulate the requests of three different service types. The traces generated are shown in Figure 5. Electricity prices, as shown in Figure 1, are the real data collected from three locations, i.e. Houston, TX, Mountain View, CA, and Atlanta, GA.

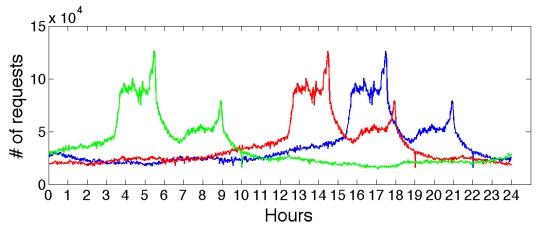
A data center's processing capacities and distances among front-end servers and data centers are generated randomly and given in Tables IV and V, respectively. Processing energy costs (Table VI) are given based on the data provided by Google's research blog [25], which are around $0.0003kWh$ for each web search request. TUFs and sub-deadlines for each type of request are collected in Table VII. Transferring costs for the three types of requests are 0.003$/mile, 0.005$/mile, and 0.007$/mile. Even though parts of the experiment setup were generated randomly, the experiment does not loss the generality. "Optimized" and "Balanced" approaches are compared.



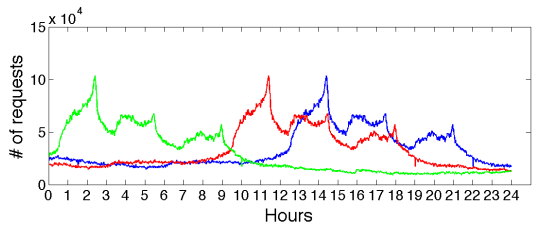
(a) Request at front-end server 1.



(b) Request at front-end server 2.



(c) Request at front-end server3.



(d) Requests at front-end server4.

Figure 5. Request traces

capacity	datacenter 1	datacenter 2	datacenter 3
$request1(\#/hour)$	3000000	3000000	3600000
$request2(\#/hour)$	3300000	3000000	3600000
$request3(\#/hour)$	3000000	3600000	4200000

Table IV
PROCESSING CAPACITIES OF EACH DATA CENTER.

B. Experimental results

1) Net profit: We first checked the net profits achieved by the two approaches. As explained in previous sections, our new model takes overall factors into consideration and provides a flexible request dispatching and resource allocation strategy to obtain a high net profit. This claim is supported by the results of running real request traces, as shown in Figure 6. Our new proposed approach ran the model once an hour (the length of a time slot is an hour).

Distance	datacenter 1	datacenter 2	datacenter 3
$front - end1(miles)$	1000	2000	1500
$front - end2(miles)$	800	1500	1000
$front - end3(miles)$	500	1000	1000
$front - end4(miles)$	1000	1500	1000

Table V
DISTANCE AMONG FRONT-END SERVERS AND DATA CENTERS.

Processing cost	datacenter 1	datacenter 2	datacenter 3
$request1(kWh)$	0.0003	0.0004	0.0006
$request2(kWh)$	0.0004	0.0003	0.0003
$request3(kWh)$	0.0006	0.0005	0.0005

Table VI
PROCESSING COST AT EACH DATA CENTER FOR DIFFERENT TYPES OF SERVICES.

It is obvious that our approach significantly outperforms the static approach in achieving a net profit.

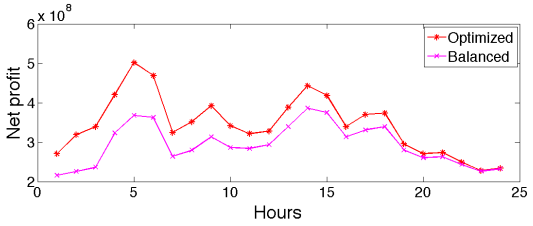


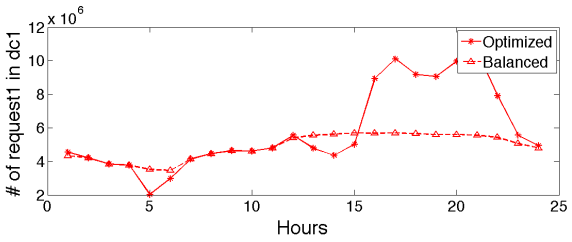
Figure 6. Net profits obtained by two approaches.

2) Request dispatching: We then studied the request dispatching of these two approaches, which are illustrated in Figure 7. From the experiment setup, we can see that for Request1, Datacenter1 and Datacenter2 have the same processing capacity, and Datacenter3 has the highest processing rate. In addition, the distances between Datacenter2 and the four front-end servers are the longest. Taking the transferring cost and processing capacities into consideration, Datacenter1 and Datacenter3 were better choices than Datacenter2 for Request1. All things considered, Datacenter2 did process some of the requests to improve the whole system's performance. However, the number of requests dispatched to Datacenter2 was still much smaller than the numbers of requests assigned to Datacenter1 and Datacenter3.

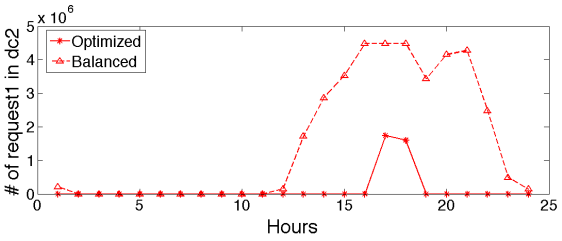
Request2 and Request3 assignments are omitted because of space limit. Similar to the allocation of Request1 shown in Figure 7, in *"Optimized"* Datacenter1 and Datacenter3 had a large part of the requests dispatched. Datacenter2 almost had no request assigned. We can see that all request dispatchings in these figures have similar results at the end of the traces in both *"Optimized"* and *"Balanced."* This explains why *"Optimized"* and *"Balanced"* had similar net profits at the end of the traces, as shown in Figure 6.

TUF	Max value	Deadline
$request1$	10 ($)	0.016 (hour)
$request2$	20 ($)	0.023 (hour)
$request3$	30 ($)	0.048 (hour)

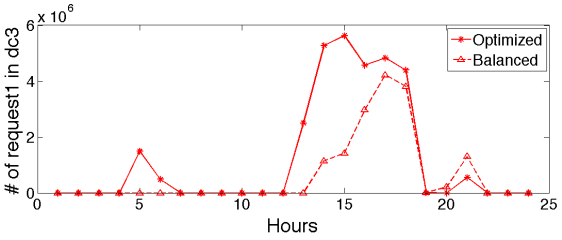
Table VII
TUFS FOR EACH TYPE OF REQUEST.



(a) Request1 allocated to datacenter1.



(b) Request1 allocated to datacenter2.



(c) Request1 allocated to datacenter3.

Figure 7. Allocations for request 1.

VII. STUDY WITH REAL TRACES USING TWO-LEVEL STEP-DOWNWARD TUFS

A. Experiment setup

Finally, we conducted an experiment to analyze computing resource allocation and request dispatching effect for a real Google workload trace that was recorded in 2010 [32]. This dataset traces over a 7-hour period. It consists of a set of tasks, where each of them runs on a single machine. We duplicated the trace and moved along time scale to simulate two different types of requests. This time we implemented two-step TUFs to represent the possible profits that may be achieved by completing requests successfully. We implemented our equation series derived in Sections IV-2 and IV-3 in both ILOG CPLEX and AIMMS (constraint logic programming solver and nonlinear programming solver.) In our experiments, we assumed that two types of requests

come from one single front-end server, and then dispatched to two data centers. There are six servers in each data center. Electricity prices for the two locations follow the prices of Houston and Mountain View, as shown in Figure 1. We selected electricity prices in the time period between 14:00 and 19:00, because the prices in that period are representative in terms of large price vibration. Processing capacities of the two types of requests in each data center were randomly generated and are shown in Table VIII. Sub-deadlines and TUF values are shown in Table IX and Table X, respectively. The power consumptions of the two types of requests in each data center are summarized in Table XI. We simply further assumed that the distances from the front-end server to those data centers are 1000 $miles$ and 2000 $miles$. The transferring costs for the two types of requests are 0.00003 and 0.00005 $\$/mile$.

Capacity	datacenter1	datacenter2
$request1(\#/hour)$	80000	70000
$request2(\#/hour)$	90000	100000

Table VIII
PROCESSING CAPACITIES OF EACH DATA CENTER.

Sub-deadline	request1	request2
$sub-deadline1$(hour)	0.001	0.01
$sub-deadline2$(hour)	0.002	0.02

Table IX
SUB-DEADLINES OF THE REQUEST.

TUF values	level1	level2	level3
$request1(\$)$	20	10	0
$request2(\$)$	30	10	0

Table X
TUF VALUES AT DIFFERENT STEPS OF THE REQUESTS.

Power	datacenter1	datacenter2
$request1$(kWh)	0.0002	0.0003
$request2$(kWh)	0.0001	0.0003

Table XI
POWER CONSUMPTION OF THE REQUESTS IN EACH DATA CENTER.

B. Experimental results

1) Net profit: Net profit achieved from the real trace using our optimized approach is shown in Figure 8. The optimized approach outperforms the balanced approach significantly. It clearly illustrates that our optimization efficiently uses electricity price difference to establish its superiority. In Figure 8, electricity price differences between Hour2 and Hour4 are larger than those at other times. The advantage of our approach is boosted at those time instances.

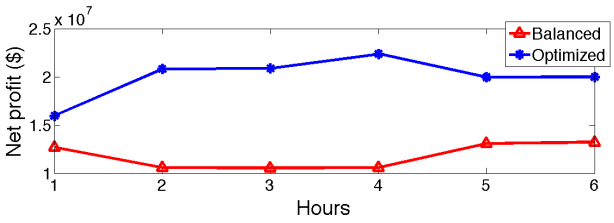


Figure 8. Net profits obtained by two approaches with two-step TUFs.

2) Request dispatching: The major difference between these two approaches comes from the request dispatching. As shown in Figure 9, it is obvious that "*Optimized*" rationally dispatched requests according to electricity prices, transferring costs, and processing capacities of each data center. All Request1 and Request2 were completed in "*Optimized*." On the contrary, 99.45% request1 and 90.19% request2 were completed in "*Balance*." Even though "*Optimized*" spent 7.74% more on the cost, it achieved a higher net profit. This observation is reasonable, since our optimization approach optimizes the trade off among several target components instead of optimizing each of them.

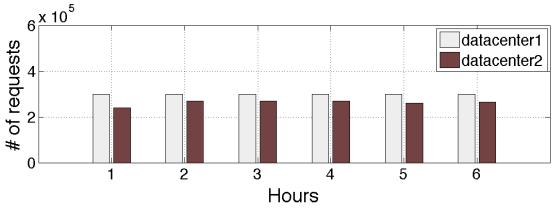
3) Workload effect: We then increased data center capacities in order to simulate a relatively low workload situation (i.e. all requests in two approaches can be completed successfully.) The result is shown in Figure 10(a). A relatively high workload situation was tested as well (i.e. no approach can complete all requests.) The result is shown in Figure 10(b). These figures prove that our optimization is superior regardless of workloads.

4) Computation time: We kept the experiment setup except changing the number of servers in each data center to service randomly generated number of service requests. We ran each server set five times and their average values were used to represent the computation times. Results are shown in Figure 11. As can be seen in the figure, the computation time increased exponentially.

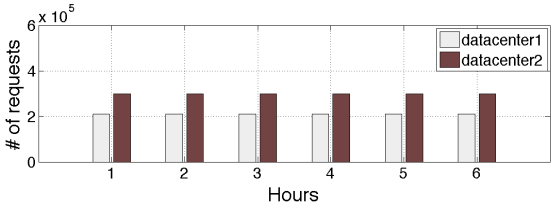
VIII. CONCLUSIONS

Cloud computing systems are proliferating. They provide increasingly diverse network-based services and applications. A large number of requests increases both the scale of data centers and their energy consumptions. Efficient request dispatching and resource allocation algorithms are in urgent need by service providers for achieving high net profits.

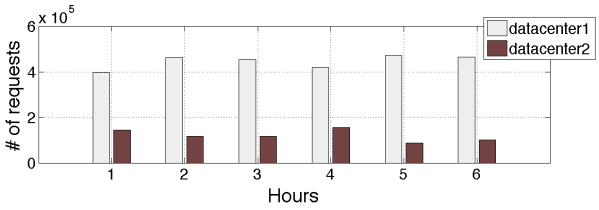
In this paper, we present an optimization-based profit- and cost-aware approach to maximize the net profits that service providers may achieve when operating distributed cloud data centers in multi-electricity-market environments. We developed a system model to effectively capture the relationship among several factors, such as SLA, cost on



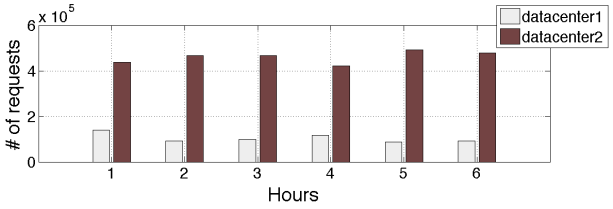
(a) Request1 allocation using balanced approach.



(b) Request2 allocation using balanced approach.



(c) Request1 allocation using optimized approach.



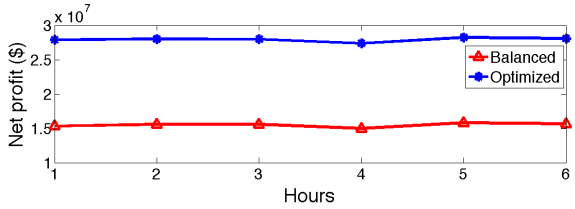
(d) Request2 allocation using optimized approach.

Figure 9. Allocations of the requests.

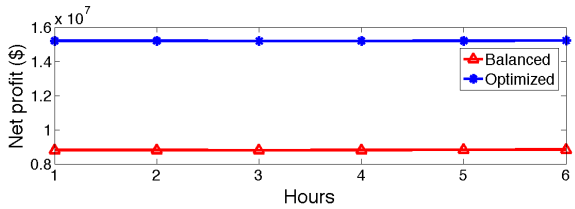
energy consumption, service request dispatching and resource allocation. One key research challenge is to formulate the problem analytically. We proposed a novel approach to transform a step-downward type of utility function to a series of well-defined constraints, so that the formulated problem can be solved with existing solvers. By considering overall factors, our approach judiciously dispatches requests and allocates computing resources. Significant net profit improvement can be achieved by efficiently using energy and computing resources. The model can be easily implemented and extended for accommodating more complex systems.

IX. Acknowledgement

This work is supported in part by NSF under projects CNS-0969013, CNS-0917021, CNS-1018108, CNS-1018731, and CNS-0746643.



(a) Net profits comparison with a relatively low workload.



(b) Net profits comparison with a relatively high workload.

Figure 10. Low/High workload situations.

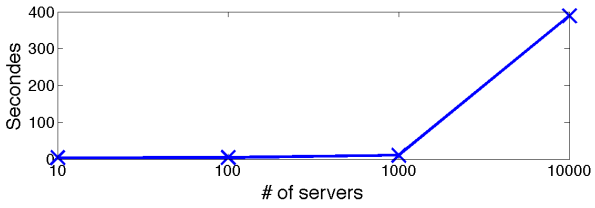


Figure 11. Computation times of different server sets.

References

[1] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. Cutting the electric bill for internet-scale systems. *SIGCOMM Comput. Commun. Rev.*, 39(4):123–134, August 2009.

[2] Lei Rao, Xue Liu, Marija Ilic, and Jie Liu. Mec-idc: joint load balancing and power control for distributed internet data centers. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, ICCPS '10, pages 188–197, New York, NY, USA, 2010. ACM.

[3] Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, and Amin M. Vahdat. Managing energy and server resources in hosting centers. In *In Proceedings of the 18th ACM Symposium on Operating System Principles SOSP*, pages 103–116, 2001.

[4] Peijian Wang, Yong Qi, Xue Liu, Ying Chen, and Xiao Zhong. Power management in heterogeneous multi-tier web clusters. In *Parallel Processing (ICPP), 2010 39th International Conference on*, pages 385 –394, sept. 2010.

[5] Zhen Liu, Mark S. Squillante, and Joel L. Wolf. On maximizing service-level-agreement profits. In *EC '01: Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 213–223, New York, NY, USA, 2001. ACM.

[6] Danilo Ardagna, Marco Trubian, and Li Zhang. Sla based resource allocation policies in autonomic environments. *J. Parallel Distrib. Comput.*, 67:259–270, March 2007.

[7] Li Zhang and Danilo Ardagna. Sla based profit optimization in autonomic computing systems. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 173–182, New York, NY, USA, 2004. ACM.

[8] Minghong Lin, Adam Wierman, Lachlan L. H. Andrew, and Eno Thereska. Dynamic right-sizing for power-proportional data centers. pages 1098–1106. IEEE, 2011.

[9] Osman Sarood and Laxmikant V. Kale. A 'cool' load balancer for parallel applications. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 21:1–21:11, New York, NY, USA, 2011. ACM.

[10] Kien Le, Ozlem Bilgir, Ricardo Bianchini, Margaret Martonosi, and Thu D. Nguyen. Managing the cost, energy consumption, and carbon footprint of internet services. In *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '10, pages 357–358, New York, NY, USA, 2010. ACM.

[11] Kien Le, Ricardo Bianchini, Jingru Zhang, Yogesh Jaluria, Jiandong Meng, and Thu D. Nguyen. Reducing electricity cost through virtual machine placement in high performance computing clouds. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 22:1–22:12, New York, NY, USA, 2011. ACM.

[12] Lei Rao, Xue Liu, Le Xie, and Wenyu Liu. Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM, 2010 Proceedings IEEE*, pages 1 –9, march 2010.

[13] E. D. Jensen, C. D. Locke, and H. Tokuda. A time-driven scheduling model for real-time systems. In *IEEE Real-Time Systems Symposium*, 1985.

[14] Haisang Wu, Umut Balli, Binoy Ravindran, and E.D. Jensen. Utility accrual real-time scheduling under variable cost functions. pages 213–219, Aug. 2005.

[15] Shuo Liu, Gang Quan, and Shangping Ren. On-line scheduling of real-time services with profit and penalty. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 1476–1481, New York, NY, USA, 2011. ACM.

[16] J. Wang and B. Ravindran. Time-utility function-driven switched ethernet packet scheduling algorithm, implementation, and feasibility analysis. *IEEE Transactions on Parallel and Distributed Systems*, 15(1):1–15, 2004.

[17] Shuo Liu, Gang Quan, and Shangping Ren. On-line real-time service allocation and scheduling for distributed data centers. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 528 –535, july 2011.

[18] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[19] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Capacity management and demand prediction for next generation data centers. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 43 –50, july 2007.

[20] Daniel Gmach, Jerry Rolia, Ludmila Cherkasova, and Alfons Kemper. Workload analysis and demand prediction of enterprise data center applications. In *Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization*, IISWC '07, pages 171–180, Washington, DC, USA, 2007. IEEE Computer Society.

[21] Shaolei Ren, Yuxiong He, and Fei Xu. Provably-efficient job scheduling for energy and fairness in geographically distributed data centers. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pages 22 –31, june 2012.

[22] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. *UC Berkeley*, 2009.

[23] Adam Nair, Jayakrishnan Wierman and Bert Zwart. Provisioning of large scale systems: The interplay between network effects and strategic behavior in the user base. under submission.

[24] Leonard Kleinrock. *Queueing Systems*, volume I: Theory. Wiley Interscience, 1975. (Published in Russian, 1979. Published in Japanese, 1979. Published in Hungarian, 1979. Published in Italian 1992.).

[25] Powering a google search@ONLINE, January 2009.

[26] Joxan Jaffar and Michael J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[27] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '87, pages 111–119, New York, NY, USA, 1987. ACM.

[28] Glpk. http://www.gnu.org/software/glpk/.

[29] Ilog. http://www-01.ibm.com/software/websphere/ilog/.

[30] Aimms. http://www.aimms.com/.

[31] M. Arlitt and T. Jin. 1998 world cup web site access logs", 1998.

[32] Joseph L. Hellerstein. Google cluster data, Jan 2010. Posted at http://googleresearch.blogspot.com/2010/01/google-cluster-data.html.