2015 IEEE 21st International Conference on Parallel and Distributed Systems

Electricity Cost Minimization in Distributed Clouds by Exploring Heterogeneity of Cloud Resources and User Demands

Zichuan Xu, Weifa Liang, and Qiufen Xia
Research School of Computer Science, Australian National University, Canberra, Australia
Email: edward.xu@anu.edu.au, wliang@cs.anu.edu.au, qiufen.xia@anu.edu.au

***Abstract*—Distributed clouds, consisting of multiple data centers located at different geographical locations, provide a plethora of services to users. They however consume enormous amounts of electricity to power their data centers. The electricity bill is almost 30% – 50% of their operational costs. Minimizing the electricity cost of distributed clouds thus is crucial to reduce the operational cost of their cloud service providers. In this paper, we study the problem of minimizing the electricity cost of a distributed cloud, by exploring the heterogeneities of cloud resources and user demands, and time-varying electricity prices, for which we first propose a two-stage optimization framework: dispatching user task requests to different data centers by incorporating the resource demands of the task requests, the workload, and the electricity price in each data center, and energy consumption profiles of different servers in each data center; followed by further energy optimization within each data center through consolidating Virtual Machines (VMs) to different servers to improve the resource utilization ratio. One critical constraint on such task dispatch and VM consolidation is to meet various user Service Level Agreements (SLAs), which include average task scheduling delays and resource demand violation limitations. Under the proposed framework, we then devise efficient scheduling algorithms for task dispatching and VM consolidations, while keeping both the average scheduling delay and resource demand violation limitation of each admitted task met. We finally evaluate the performance of the proposed algorithms through experimental simulations, using real data sets - the real electricity prices and task traces. Experimental simulation results demonstrate that the proposed algorithms are promising.**

I. Introduction

Distributed clouds, consisting of multiple data centers located at different geographical locations, are emerging as the new cloud computing paradigm recently. Although distributed clouds can provide cheap and unprecedented-scale compute and storage resources to meet various user demands, they are big electricity consumers, and the electricity bill dominates the expenditure of the entire operation of each data center, i.e., 30%-50%, and this figure is almost doubled every five years [12], [15]. Consider that electricity prices in different regions fluctuate over time, and servers in each data center exhibit the heterogeneity in terms of energy consumptions as they come from different vendors and/or different generations of the same vendor [23], [24], [30]. This creates an opportunity to minimize the electricity bill of cloud service providers by dispatching user task requests to the data centers that have not only cheaper electricity prices [22], [26], [28] but also available servers with less energy consumptions. On the other hand, task requests are heterogeneous in terms of durations, resource demands, and Service Level Agreements (SLAs). For example, many enterprises have office automation applications with fast response time requirements for their employees during working time, and backup and analyzing task requests with strict resource demand violation limitations for their massive historical data during off-work time. Along with the time-varying electricity prices and the heterogeneous servers, such heterogeneity of task requests bring lots of challenges in dispatching task requests to different data centers and task allocation to different servers within each data center. For example, how to dispatch task requests to different data centers while jointly considering the energy consumptions of the task requests, workloads and electricity prices of data centers, how to consolidate the task requests with complementary resource demands to servers to promote server utilizations and reduce server energy consumptions, and how to guarantee the SLAs of admitted task requests, e.g., their average scheduling delays and the resource demand violation limitations are met.

In this paper we will address the mentioned challenges by exploring time-varying electricity prices and the heterogeneities of cloud resources and user demands in a distributed cloud. We aim to minimize the electricity cost of the distributed cloud, by dispatching task requests to data centers with cheaper electricity prices and their servers have less energy consumptions, and consolidating the dispatched task requests with complementary resource demands within each data center further, while meeting various user SLAs. Despite that extensive studies on reducing energy consumption of data centers in literature have been conducted in the past several years, a few attentions have been paid to explore the heterogeneities of cloud resources and user demands in distributed clouds. Most existing studies focused only on either homogeneous cloud resource [22], [26], [28] or homogeneous task requests [5], [17], [22], [26], [28] in a single data center [14], [19], [30]. None of them explores resource provisioning with complementary resource demands [1], [6], [24], [29], [30]. Furthermore, almost all these mentioned studies assumed only one single SLA requirement: either the average schedule delay or the resource demand violation limitation. Unlike these existing works, in this paper we aim to minimize the electricity cost in a distributed cloud, through exploring the heterogeneities of both cloud resources and user demands, utilizing the time-varying electricity prices by dispatching task requests to different data centers, and consolidating the VMs with complementary resource demands within each data center, while keeping the multiple SLA requirements (e.g., the average schedule delay and resource demand violation limitation) met. To the best of our knowledge, we are the first to jointly consider the heterogeneous cloud resources and user demands, the multiple SLA requirements of users, and the complementary resource demands of different task requests, in the resource allocation to meet different user demands.

The main contributions of this paper are as follows. We first propose an optimization framework for task scheduling to minimize the electricity cost, by exploring the heterogeneities of both cloud resources and user demands. The framework consists of two phases: dispatching task requests to different

1521-9097/15 $31.00 © 2015 IEEE
DOI 10.1109/ICPADS.2015.56

data centers according to electricity prices, workloads of different data centers, and the expected energy consumptions of task implementation in the dispatched data centers, followed by consolidating the virtual machines (VMs) for task requests with complementary resource demands to further reduce the electricity cost. We also evaluate the effectiveness of the proposed framework and the performance of the proposed algorithms, using real electricity price and task trace data sets. Experimental results demonstrate that the proposed algorithms are promising.

The remainder of this paper is organized as follows. Section II introduces related work, followed by the system model and problem definition in Section III. Section IV presents the optimization framework. Section V evaluates the effectiveness of the framework and the performance of the proposed algorithm through experimental simulations. The conclusion is given in Section VI.

II. Related Work

Promoting energy efficiency of data centers is one efficient approach to reduce the electricity bills of data centers. Extensive studies along this line have been taken in the past, e.g., homogeneous servers [5], [17], diverse electricity prices in different data center locations [22], [24], and making use of different renewable power sources [17]), interactive user requests [24], [26], [28], batch tasks [1], MapReduce tasks [29], virtual network requests [27], and a mix of them [6], [16], [24], [29]. For example, Lin *et al.* [17] observed that much energy is wasted in maintaining the excess service capacity during low load periods, and they devised an online algorithm for minimizing the energy consumption by dynamically resizing active servers in a data center. Qureshi *et al.* [22] initialized the electricity cost minimization problem in data centers by exploring fluctuating electricity prices at different data centers. Following their work, research on geographical load balancing in a distributed cloud with multiple data centers under renewable energy sources has been conducted [18]. On the other hand, with more and more users outsourcing their IT services to clouds, data centers are keeping upgrading their services to the state-of-the-arts, thereby leading to the heterogeneity of servers. Cloud applications in a data center are also becoming more diverse in terms of resource demands.

Several studies explored the heterogeneities of both cloud resources and user demands in a single data center [6], [19], [29], [30]. For example, Garg *et al.* [6] studied the resource provisioning problem for heterogeneous workloads through jointly scheduling interactive web requests and batch tasks. Following the same line of research, Zhan *et al.*[29] considered multiple types of workloads within a data center, i.e., batch tasks, web server requests, search engine tasks, and MapReduce tasks. Zhang *et al.* [30] considered multiple types of users tasks by applying the K-means clustering algorithm to classify user tasks into distinct classes, and proposed a method to schedule various user tasks, by dynamically adjusting the number of servers for different types of VMs requested by the tasks. They however did not consider the consolidation of VMs with complementary resource demands. There are also a few studies exploiting the heterogeneities of both cloud resources and user demands in distributed clouds too [1], [24]. For example, Ren *et al.* [24] considered the heterogeneities of both servers and user tasks, by proposing an online task scheduling algorithm to minimize the energy cost, while maintaining the rate fairness among different sources. Adnan *et. al.* [1] formulated the problem of scheduling various workloads (interactive requests and batch tasks) to geographically distributed data centers by providing a mixed linear integer programming solution. Some other studies focus on designing task classification algorithms [20], [23]. However, none of them considered the consolidation of VMs with complimentary resource demands within each data center. In addition, most of them only considered one aspect of user SLAs, i.e., either the average scheduling delay [24], [26], [28], [30] or the resource demand violation limitation [19], not both.

In addition to exploring all above mentioned properties of data centers and tasks, another key factor for achieving energy efficiency in a distributed cloud is resource provisioning, which refers to allocating resources to VMs to match their workloads. While under-provisioning degrades application performance and may lose customers, over-provisioning leads to high operational costs by keeping running idle servers. There are several studies focusing on consolidation of VMs [14], [19]. They however only considered consolidation of a pair of two VMs for a data center with homogeneous servers. To the best of our knowledge, we are the first to explore the heterogeneity of both cloud resources and user demands in a distributed cloud, while jointly consider further electricity cost reduction within each data center by consolidating VMs with complementary resource demands.

III. Preliminaries

A. System model

We consider a distributed cloud $G = (\mathcal{DC} \cup \mathcal{WP}, E)$ consisting of a number of data centers and web portals at different geographical locations, where $\mathcal{DC}$ and $\mathcal{WP}$ are the sets of data centers and web portals, E is the set of high speed links between web portals and data centers. Each web portal $WP_m \in \mathcal{WP}$ accepts task requests by its nearby users, while each data center $DC_i \in \mathcal{DC}$ contains different types of servers with different performance and energy consumption profiles. Assume that the servers in each data center are classified into different types according to their hardware configurations. Let K be the number of server types in each data center. Similarly, different task requests have different resource demands and execution durations, thereby incurring different cloud resource and energy consumptions [6], [16], [20], [23], [29], [30]. To serve a wide range of task requests, cloud service providers offer different types of VMs to meet the resource demands of different task requests. For example, Amazon EC2 has over 20 types of VMs with each type having a different quantities of resources [2]. We thus classify task requests and VMs into Q and J types, respectively.

We consider a distributed cloud operating for a quite long period (e.g., one month, or one year). Assuming that time is slotted into equal time slots, the resource demands of each task request are estimated at each time slot t during the task execution duration. Let y_q be a type-q task request, denote by $\mu_r(y_q, t)$ and $\sigma_r(y_q, t)$ the mean and standard variation of the resource demands of y_q at time slot t, which all are given as a *priori*, $1 \leq q \leq Q$. As type-q task requests can be executed by different types of VMs, we assume that each type-j VM is capable of executing all types of task requests, where $1 \leq j \leq J$. Following existing studies [25], [30], we further assume that each VM serves one task request each time. That is, for type-j VMs, there is a $M/G/N_{j,i}(t)$ waiting queue [9] for each type of task requests, where $N_{j,i}(t)$ is the number of available type-j VMs in data center DC_i at time slot t.

B. User SLAs with multiple requirements

The SLA of a task request consists of multiple aspects. In this paper, we assume that each SLA consists of the average scheduling delay requirement, and the resource demand violation limitation.

The average scheduling delay requirement: recall that each type of task requests in a $M/G/N_{j,i}(t)$ waiting queue of type-j VMs is waiting to be processed by $N_{j,i}(t)$ type-j VMs in data center DC_i at time slot t. Meeting the average scheduling delay requirement of such a task request is to limit its average waiting delay. Let D_q be the average scheduling delay requirement of type-q task requests, and $D^q_{j,i}$ the average scheduling delay experienced by type-q task requests that will be processed by type-j VMs at DC_i. Thus,

$$D^q_{j,i} \leq D_q. \tag{1}$$

$D^q_{j,i}$ can be calculated by [9]

$$D^q_{j,i} = \frac{1}{u_{j,i}N_{j,i}(t) - \lambda^q_{j,i}(t)} \frac{1+CV_q^2}{2}, \tag{2}$$

where $u_{j,i}$ is the service rate of type-j VMs at data center DC_i, $\lambda^q_{j,i}(t)$ is the arrival rate of type-q task requests that will be processed by type-j VMs at data center DC_i, and CV_q is the coefficient of variation of durations of type-q task requests defined as the standard deviation divided by the mean.

The resource demand violation limitation: users specify their tolerable numbers of resource demand violations during a *SLA monitoring period* [2], [19], which will be examined periodically at the end of each SLA monitoring period. Let T be the number of time slots in a SLA monitoring period. The resource demand violation limitation of each type-q task request during a SLA monitoring period is defined as [19]

$$E\Big[\sum_{t=0}^{T} \Phi(\text{ demands exceed supplies at time slot } t)\Big] \leq \beta_q, \tag{3}$$

where β_q is the tolerable number of resource demand violations for task request y_q during period T with $0 \leq \beta_q \leq T$, $\Phi(s) = 1$ if s is true; and 0 otherwise. This resource demand violation limitation metric guarantees that the expected number of resource demand violations in period T should be no greater than a specified threshold β_q. We assume that the violation of resource demand at each time slot is independent with each other, since resource demand violations may be caused by irrelevant events, such as lack of resources, unexpected bursty demands, etc.

C. Problem definition

Given a distributed cloud $G = (\mathcal{DC} \cup \mathcal{WP}, E)$, each $DC_i \in \mathcal{DC}$ consists of clusters of heterogeneous servers, while each cluster $S_{k,i}$ contains a number of type-k homogeneous servers with $1 \leq i \leq N$ and $1 \leq k \leq K$. Assuming that the arrival rate of each type of task requests at each web portal and a monitoring period T are given, the *electricity cost minimization problem in the distributed cloud* G is to assign each DC_i with a fraction of the aggregated task request arrival rate from all web portals at each time slot such that the *electricity cost* of implementing admitted task requests in G is minimized within the monitoring period, subject to that both the average scheduling delay requirement and the resource demand violation limitation are met within the monitoring period. The electricity cost of a data center is determined by its local electricity price and the amount of energy its servers consumed, where the energy consumption of each server is proportional to its computing resource utilization [24], [26], [27], [28]. We further assume that the electricity prices do not change within a time slot, and let $p_i(t)$ be the electricity price of data center DC_i at time slot t. The electricity cost of data center DC_i during the monitoring period T is then given by

$$E(DC_i, T) = \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{m=1}^{|S_{k,i}|} P_k \cdot r_{m,k} \cdot p_i(t), \tag{4}$$

where $|S_{k,i}|$ denotes the number of servers in cluster $S_{k,i}$, P_k is the peak power of a server in the cluster, and $r_{m,k}$ is the computing resource utilization of the mth server in the cluster. Notice that the computing resource utilization ratio $r_{m,k}$ of the mth server in $S_{k,i}$ is determined by the task requests that are allocated to the server.

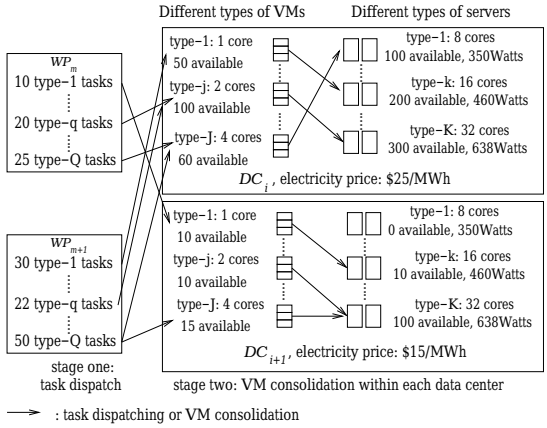


Fig. 1: An example of the proposed framework

IV. An Optimization Framework

Consider the heterogeneities of both cloud resources and user demands, in this section we present an optimization framework for the electricity cost minimization problem in the distributed cloud G. We start by giving an overview. We then show how to map various task requests from different web portals to different data centers. We finally show how to consolidate some VMs in a data center to further reduce the electricity consumption.

A. Overview

To minimize the electricity cost of the distributed cloud, the proposed framework adopt a reduce-and-refine strategy: the first stage reduces the electricity cost by dispatching task requests from different web portals to different data centers based on the time-varying electricity prices, workloads and VM energy consumptions of the data centers. The second stage further reduces the electricity cost by consolidating some VMs with complementary resource demands within each data center, as illustrated by an example in Fig. 1.

Recall that each task request is executed in a single VM. This means the energy consumption of a task request is actually the energy consumption of the VM. Such energy consumptions of VMs within a given monitoring period can be predicted, following an autoregressive moving average process with given weights that are related to their energy consumptions in previous time slots [11]. Thus, dispatching task requests to different data centers is to dispatch task

requests to different types of VMs with predicted energy consumptions at different data centers. Given the dispatched task requests at each data center, the electricity costs by executing the task requests can be further reduced by promoting server utilizations in the data center, since high utilizations can reduce energy wastage in idle servers. To this end, the dispatched task requests will be consolidated by their complementary resource demands in each data center. **Algorithm** 1 gives a brief description of this framework.

Algorithm 1 *An optimization framework for the electricity cost minimization problem in $G = (\mathcal{DC} \cup \mathcal{WP}, E)$ at time slot t*

1: /* **Stage one: task request dispatch from web portals to data centers** */
2: For each data center DC_i, predict the energy consumptions by its VMs of each type in the future monitoring period T by autoregressive moving average predictions;
3: Dispatch task requests at all web portals to different data centers by invoking **Procedure** TaskDispatch;
4: /* **Stage two: Consolidation of VMs with complementary resource demands within each data center** */
5: **for** each data center $DC_i \in \mathcal{DC}$ **do**
6: Allocate each dispatched task request to a VM of its assigned type, then call **Procedure** VMConsolidation;
7: According to VM consolidation results, log the real energy consumption of each type of VMs for the energy consumption prediction in future;

B. Task request dispatch to different data centers

The main focus of stage one, task request dispatch, is to deal with which types of task requests from different web portals should be dispatched to which data center. To this end, we reduce the task request dispatch problem to a minimum cost multi-commodity flow problem in an auxiliary graph $G_f = (V_f, E_f; u, c)$ as follows, where edge costs $c : E_f \mapsto \mathbb{R}^{\geq 0}$ represent the costs of routing one unit of a commodity through the corresponding edges corresponding to the electricity cost of a task request, and edge capacities $u : E_f \mapsto \mathbb{R}^+$ denote the number of units that can be routed through the edges, which implies the SLA requirement of each task request. G_f is constructed such that each type of task requests in each web portal is considered as a commodity with one task request corresponding to one unit of the commodity, and routing the commodities of all web portals in G_f is equivalent to dispatching task requests to different data centers.

There are four types of nodes in V_f of G_f:

- **source nodes and a virtual sink:** for each commodity with type-q task requests at web portal WP_m, there is a *source node* $s_{q,m}$ in G_f where the commodity originates, and a *virtual sink* t_0 where the commodity should be routed
- **queue nodes:** for each data center DC_i, there is a type-q *queue node* for each type-j VM node denoting that type-q task requests may be executed by type-j VMs
- **VM nodes:** for each data center DC_i, there is a type-j *VM node* representing the VMs of each type j in DC_i
- **data center nodes:** each denotes a data center.

There are edges in E_f from each source node $s_{q,m}$ to type-q queue nodes in the data centers, from type-q queue nodes to their corresponding type-j VM nodes, from type-j VM nodes to DC_i, and from all data centers to the virtual sink t_0.

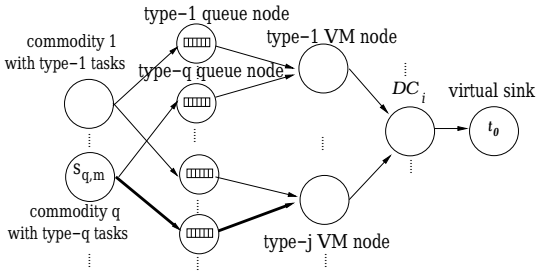


Fig. 2: $G_f = (V_f, E_f; u, c)$

As illustrated by the highlighted path in Fig. 2, a route from each $s_{q,m}$ to t_0 implies that a fraction of type-q task requests at web portal WP_m will be dispatched to a data center, say DC_i, by first waiting in a queue for their type and then processed by a type of VMs.

The capacities of edges in G_f is set as the average scheduling delay requirement. Intuitively, to meet the average scheduling delay requirement, the number of task requests waiting at each type-q queue node should be limited. We thus set the capacity of the edge from each type-q queue node to each type-j VM node as the maximum number of task requests that can be served without violating their delay requirements. Let $u_t^{(i)}(q, j)$ be the capacity of the edge from a type-q queue node to a type-j VM node in DC_i. By Eqs. (1) and (2), we then have

$$u_t^{(i)}(q, j) = \lfloor u_{j,i} N_{j,i}(t) - \frac{1 + CV_q^2}{2D_q} \rfloor. \tag{5}$$

Recall that $N_{j,i}(t)$ is the number of all available type-j VMs at time slot t, Eq. (5) means that if all these $N_{j,i}(t)$ type-j VMs execute type-q task requests, the number of the 'admissible' type-q task requests of type-j VMs will reach the maximum. However, type-j VMs may execute the other types of task requests as well. Thus, the capacity of each type-j VM node in DC_i is

$$u_t^{(i)}(j, i) = \lfloor u_{j,i} N_{j,i}(t) - \frac{1 + CV_{max}^2}{2D_{min}} \rfloor, \tag{6}$$

where D_{min} and CV_{max} are the minimum delay requirement and maximum CV of all types of task requests. Here, $N_{j,i}(t)$ can be strategically set to admit more task requests while keeping the electricity cost minimized. We adopt a greedy approach to set $N_{j,i}(t)$. Specifically, $N_{j,i}(t)$ is in proportion to the ratio of the number of task requests executed by type-j VMs in DC_i to the total electricity cost incurred by type-j VMs at time slot $t-1$. The capacities on other edges are set to the accumulative task request arrival rate from all web portals.

The costs of edges in E_f are set to the electricity costs of routing one task request through the corresponding edges. Consider a type-q task request, the electricity cost of executing it at DC_i in the monitoring period T is determined by the electricity price at DC_i and the energy consumption of its allocated VM within the monitoring period. We thus set the cost of the edge between each type-q queue node in DC_i to its type-j VM node as the electricity cost incurred by a type-q task request in the monitoring period T. Denote by $c_t^{(i)}(q, j)$ the cost of the edge, then

$$c_t^{(i)}(q, j) = \sum_{t'=0}^{T-1} p_i(t + t') \cdot \varepsilon_{j,i}(t + t'), \tag{7}$$

where $\varepsilon_{j,i}(t + t')$ is the predicted energy consumption by a

type-j VM in data center DC_i at time slot $(t+t')$ after the execution of step 2 in algorithm 1. The costs on other edges are set to zeros.

Having constructed the flow graph G_f, dispatching task requests from web portals to data centers then is reduced to finding a minimum-cost multi-commodity flow in G_f with sink node t_0. The detailed algorithm for dispatching task requests is shown by **Procedure** TaskDispatch.

Procedure TaskDispatch()

Input: The distributed cloud $G = (\mathcal{DC} \cup \mathcal{WP}, E)$, and the task request rates arrived at each web portal $WP_m \in \mathcal{WP}$

Output: Task requests dispatched at each data center $DC_i \in \mathcal{DC}$

1: Create an empty flow graph G_f, and consider each queue, each VM type and each data center in Fig. 2 as nodes in G_f;
2: Add a starting node $s_{q,m}$ for each task type q at each web portal WP_m to G_f, and a virtual sink node t_0 to G_f;
3: There is an edge from each $s_{q,m}$ to each type-q queue node in each data center, an edge from a queue node to its VM type as shown in Fig. 2, an edge from each VM node to its data center, and an edge from each data center node to t_0;
4: Set edge capacities and costs by Eqs. (5), (6) and (7);
5: Find a minimum cost multi-commodity flow in G_f for all commodities with sink node t_0;
6: Allocate task requests to each data center DC_i according to the flow routed on the edge from DC_i to t_0;

C. Task request allocation within a data center by VM consolidation

Once a task request has been dispatched to a data center, a VM of its allocated type will be assigned to the task request. To further improve resource utilizations of the data center thereby reducing the electricity cost, all assigned VMs with complementary resource demands in the data center are consolidated if possible. To this end, we first introduce a novel *VM correlation model* to model the complementary resource demands among the VMs, we then group the VMs to different groups with each having highly-complementary VMs, and allocate groups of VMs to different servers.

We start with the VM correlation model. Two highly-correlated VMs have complementary resource demands and thus least coincided demand peaks or valleys. If they are consolidated into a single server, they are more likely to share resources with each other, thereby leading to less resource demand violations. Recall that the resource demand violation requirement of each task request is enforced within each SLA monitoring period. We thus model the demand correlation of two VMs within a future SLA monitoring period starting at time slot t. Let m_1 and m_2 be two VMs for task requests y and y', respectively. Intuitively, highly-correlated VMs have pairs of matched demand peak and valley at many time slots of the SLA monitoring period. They as a whole thus have a nearly 'smooth' demand curve in the SLA monitoring period that consists of T time slots. Let $\rho(m_1, m_2, t)$ be the correlation between m_1 and m_2 at time slot t, and recall that the type-r resource demands by task requests y and y' are estimated by their means $\mu_r(y,t)$ and $\mu_r(y',t)$. $\rho(m_1, m_2, t)$ thus can be calculated by

$$\rho(m_1, m_2, t) \approx \frac{\sum_{t'=t+1}^{t+T}(\mu_r(y,t') + \mu_r(y',t'))}{T \cdot \max\limits_{t'=t+1,t+2,\ldots,t+T}\{\mu_r(y,t') + \mu_r(y',t')\}} \cdot \frac{1}{T}, \tag{8}$$

The first term in the right hand side of Eq. (8) will approximate T if m_1 and m_2 have perfectly-matched pairs of demand peaks and valleys, i.e., 'smooth' demand curves. Thus, $\rho(m_1, m_2, t) \approx 1$ means that m_1 and m_2 are highly-correlated, and should be consolidated to a single server.

A correlation graph $G_c^i = (V_c, E_c)$ for all VMs allocated to data center DC_i is then constructed, where each node in V_c represents a VM, and each edge in E_c denotes a correlation between its corresponding two VMs with weight $\rho(m_1, m_2, t)$. G_c^i is partitioned into several subgraphs with each denoting a group of VMs to be consolidated to a single server. The maximum size of each sub-graph (the number of VMs it contains) plays a non-trivial role in VM consolidation. Intuitively, the larger the size a subgraph, the more the number of VMs will be consolidated in a single server. By doing so however may reduce the number of admitted VMs, because servers in each cluster have only limited available resources and very likely reject large-size subgraphs. Thus, the maximum size of each subgraph, δ, is set to the mean number of VMs with the least processing rate that can be admitted by all clusters. To partition G_c^i into subgraphs with size no more than δ, an efficient partition algorithm in [13] is applied.

We then allocate the VMs in each subgraph G_{sub} to a cluster within DC_i and then to a single server in the allocated cluster, such that their resource demands are satisfied. An exact satisfaction of the resource demands by the VMs in G_{sub} is impossible, since the exact resource demands are hard to be obtained. Instead, each G_{sub} is allocated with an amount that preserves the resource demand violation limitations of all VMs in G_{sub}. With a little abuse of notations, $x_r(G_{sub}, t)$ is used to represent the amount of type-r resource that should be allocated to G_{sub}, and it is calculated by

$$x_r(G_{sub}, t) = \sqrt{\frac{\sigma_r(G_{sub}, t)}{\beta_{min}}} + \mu_r(G_{sub}, t). \tag{9}$$

where β_{min} is the minimum resource demand violation requirement of all task requests that run in the VMs in G_{sub}, $\mu_r(G_{sub}, t)$ and $\sigma_r(G_{sub}, t)$ are the mean and variation of demanded type-r resources by all VMs in G_{sub}. The correctness of Eq. (9) is analyzed in Theorem 1. Given the amount of resources that should be allocated to the VMs in G_{sub}, G_{sub} is first allocated to a cluster and then to a server within the cluster, by transforming the VM consolidation problem to a weighted maximum bipartite matching problem. Detailed procedure is summarized in **Procedure** VMConsolidation.

D. Algorithm Analysis

We first analyze the time complexity of **Algorithm** 1 by analyzing the running times of its two procedures, i.e., **Procedure** TaskDispatch and **Procedure** VMConsolidation, in lemmas 1 and 2. We then analyze its correctness and the time complexity in Theorem 1.

Lemma 1: Given a distributed cloud $G = (\mathcal{WP} \cup \mathcal{DC}, E)$, arrival rates of Q types of task requests at each web portal in $\mathcal{WP}$, J types of VMs at each data center in $\mathcal{DC}$, algorithm TaskDispatch takes $O^*(\epsilon^{-2}Q^2J^2|\mathcal{WP}|^2|\mathcal{DC}|^2)$ time[1] to dispatch various task requests from different web portals to different data centers, if the Garg and Könemann's algorithm [7] is applied, where ϵ is the accuracy parameter of Garg and Könemann's algorithm with $0 < \epsilon \leq 1/3$.

Proof: The auxiliary flow network $G_f = (V_f, E_f; u, c)$ consists of $|V_f| = Q \cdot |\mathcal{WP}| + Q \cdot J \cdot |\mathcal{DC}|$ nodes and $|E_f| = Q \cdot J \cdot |\mathcal{WP}| \cdot |\mathcal{DC}| + Q \cdot J \cdot |\mathcal{DC}|$ edges. Thus, the construction of G_f takes $O(|V_f| + |E_f|)$ time. Applying the

[1] $O^*(f(n)) = O(f(n)\log^{O(1)} n)$

Procedure VMConsolidation()

Input: Data center DC_i and the task requests that are dispatched to it at time slot t

Output: VM consolidation results

1: Build a correlation graph $G_c^i = (V_c, E_c)$ for all task requests dispatched in DC_i, where each node in V_c represents a VM and its corresponding task request, each edge in E_c denotes a correlation between two VMs, and the weight on each edge is set to $\frac{1}{\rho(m_1,m_2)}$;
2: Let $N_{k,i}$ be the number of available servers in each cluster $S_{k,i}$;
3: $\delta \leftarrow \frac{\sum_k N_{k,i}(t)\cdot u_{k,i}}{K\cdot \min_j u_{j,i}}$, where $u_{k,i}$ is the server processing rate in $S_{k,i}$;
4: Partition G_c^i into $\lceil |V_c|/\delta \rceil$ subgraphs with each size no more than δ;
5: Create a set of *virtual clusters* with each representing a portion of available computing resources in its cluster, the size of these virtual clusters is $\delta \cdot \max_j u_{j,i}$;
6: Build a bipartite graph, where vertex set on one side is the set of subgraphs, and that on the other is a set of virtual clusters;
7: Set the weight of each edge from one side to the other as the potential electricity cost incurred in the cluster;
8: Find a weighted maximum matching which corresponds an allocation of the sub-graphs to different clusters in DC_i;
9: **for** Each cluster $S_{k,i}$ of data center DC_i **do**
10: Calculate the amounts of resources that should be allocated to G_{sub} by Eq. (9);
11: Find a server in $S_{k,i}$ for G_{sub} to meet its demand (Eq. (9));
12: If there are still un-assigned sub-graphs, further divide each sub-graph into smaller sizes, until no sub-graphs can be admitted by servers in $S_{k,i}$;

algorithm by Garg and Könemann in G_f to find a minimum-cost multi-commodity flow takes $O^*(\epsilon^{-2}m(n+m))$ time to deliver an approximate solution with approximation ratio of $(1-3\epsilon)$ while the associated cost is minimized, where $m = |E_f|$ and n is the number of commodities [7]. Recall that there is a source node $s_{q,m}$ in G_f for each type of task requests at each web portal, we thus have $Q \cdot |\mathcal{WP}|$ commodities that need to be routed to the virtual sink t_0 in G_f. That is, $n = Q \cdot |\mathcal{WP}|$. Thus, procedure TaskDispatch takes $O^*(\epsilon^{-2}Q^2J^2|\mathcal{WP}|^2|\mathcal{DC}|^2)$ time. ■

Lemma 2: Given a distributed cloud $G = (\mathcal{WP} \cup \mathcal{DC}, E)$ and the correlation graph $G_c^i = (V_c, E_c)$ of dispatched task requests at each data center $DC_i \in \mathcal{DC}$, the time complexity of applying **Procedure** VMConsolidation in DC_i is $O(|V_c|^3 \log |V_c|)$.

Proof: **Procedure** VMConsolidation consists of three stages: (1) constructing and partitioning correlation graph G_c^i; (2) matching subgraphs to different clusters; and (3) packing subgraphs to servers in each cluster. By applying a partitioning algorithm in [13], part (1) takes $O(n^3(\log n - \log \delta)) = O(n^3 \log n)$ to partition G_c^i into several subgraphs with a maximum size δ, where n is the number of VMs in G_c^i, i.e., $n = |V_c|$. Recall that K is the number of clusters in each data center DC_i. The time complexities of the matching and packing in parts (2) and (3) are $O((n/\delta+K)^3) = O((n+K)^3)$ and $O((n/\delta)^2) = O(n^2)$, respectively, if $\delta = 2$ in the worst case. Note that $O((n+K)^3) = O(n^3)$ if $K \ll n$. Thus, the time complexity of procedure VMConsolidation is $O(n^3 \log n) = O(|V_c|^3 \log |V_c|)$. ■

Theorem 1: Given a distributed cloud $G = (\mathcal{WP} \cup \mathcal{DC}, E)$ and various types of task requests arrived at each web portal in $\mathcal{WP}$ with each type-q task requests having an arrival rate of $\lambda_j^q(t)$ at time slot t, there is an algorithm takes $O(\epsilon^{-2}Q^2J^2|\mathcal{WP}|^2|\mathcal{DC}|^2 \log^{O(1)} QJ|\mathcal{WP}||\mathcal{DC}| + |\mathcal{DC}|\lambda_j^q(t)^3 \log \lambda_j^q(t))$ to deliver a feasible solution, in which both the average scheduling delay and resource demand violation limitation requirements of each admitted task request are met.

Proof: Following Lemmas 1 and 2, the running times of **Algorithm 1** is the total time spent in one invoking of **Procedure** TaskDispatch and $|\mathcal{DC}|$ times invoking of **Procedure** VMConsolidation.

For the correctness, we first show that the average scheduling delay requirement of each type-q task request is met. This is to show that the experienced average waiting time $D_{j,i}^q$ of each queue for type-q task requests in each data center DC_i is within the required delay D_q, i.e., $D_{j,i}^q \leq D_q$ (Eq. (1)). Given the constructed flow graph G_f and its capacity settings (Eqs. (5) and (6)), we know that the *bottleneck capacity* is the capacity of the edge from each type-j VM node to the data center node DC_i, i.e., Eq. (6). This means that the number of admitted task requests by all queues of type-j VMs in DC_i is limited by this bottleneck capacity, since the maximum flow in the edge will not exceed its capacity by the minimum-cost multi-commodity flow problem. By Eq. (6), the bottleneck capacity is calculated by the minimum delay requirement D_{min} and the maximum CV_{max} of all types of task requests. According to Eqs. (2) and (6), the experienced delay $D_{j,i}^q$ of task requests of each type q is no greater than D_{min}. That is,

$$D_{j,i}^q \leq D_{min}. \tag{10}$$

As $D_q \geq D_{min}$, we have $D_{j,i}^q \leq D_q$. Thus, the average delay requirement of each task request is met.

We then show the resource demand violation limitation of each type-q task request admitted by data center DC_i is also satisfied. Recall that within each data center DC_i type-r resources are allocated to a group G_{sub} of VMs with highly correlated resource demands by the amount $x_r(G_{sub}, t)$, i.e., Eq. (9). We thus only need to show that the resource demand violation requirement of each VM in G_{sub} is met under this allocated amount, i.e.,

$$E[\sum_{t=0}^{T} \Phi(v_r(G_{sub}, t) > x_r(G_{sub}, t))] \leq \beta_q, \tag{11}$$

where $v_r(G_{sub}, t)$ is the real amount of type-r resource required by VMs in G_{sub}. By Chebyshev's inequality [21] and Eq. (9), we have

$$\begin{aligned} &E[\sum_{t=0}^{T} \Phi(v_r(G_{sub}, t) > x_r(G_{sub}, t))] \\ &\leq \frac{\sigma_r(G_{sub}, t)}{(x_r(G_{sub}, t) - \mu_r(G_{sub}, t))^2} = \beta_{min} \leq \beta_q. \end{aligned} \tag{12}$$

The theorem holds. ■

V. Experimental Study

In this section we evaluate the effectiveness of the proposed framework by experimental simulations, using real-life electricity price and task trace data sets.

A. Simulation environment

We consider a distributed cloud consisting of five data centers and six web portals. The five data centers are located at Council Bluffs (IA), Houston (TX), Boston (MA), Mountain View (CA) and Lenoir (NC) in the States. Following the similar settings in [20], [23], [30], each data center hosts 5,000 heterogeneous servers consisting of three different platforms

A, B, and C, as shown in Table I. The servers at each data center are organized into different types of clusters by different combinations of memory and compute capacities, as illustrated in Table I. Each cluster consists of two server racks with each having 42 servers. There are 4 types of VMs with each having a different number of cores ranged from 1 to 4, i.e., $J = 4$. The length of each time slot is 5 minutes, and the length of a SLA monitoring period is one hour.

TABLE I: CPU, memory and storage capacities [20], [23], [30]

Cluster type	Platform*	CPU (Cores)	Memory (GB)	Storage (GB)	Active power (Watts)	Idle power (Watts)
1	A	8	8	62.5	350	210
2	B	16	16	125	460	270
3	B	16	8	125	460	270
4	B	16	24	125	460	270
5	B	16	4	125	460	270
6	C	32	32	250	638	380

*Each platform refers to a micro-architecture and memory combination [20], [23], [30]

Following the similar settings in existing studies [26], [28], the average electricity price per hour at each data center is derived from the raw data available from US government agencies [3], [4], and the prices at all data centers are shifted according to their time zones.

We adopt real Google task request traces as the task request arrival sequence [8], [20], [30]. According to [8], [20], [30], these task requests are classified into 8 types as shown in Table II, i.e. $Q = 8$. Most of these task requests are short-term task requests [8], [20], [30] with duration of less than 2 hours. In our evaluation, around 80 percent of task requests are types 1 to 4, while the rest 20 percent are types 5 to 8. The minimum, maximum and CV of memory and storage demands by each type of task requests are provided. Note that some of the values in Table II are derived from figures in [20], [30], which may not be real values but are accurate enough to characterize each type of task requests. The resource demands of each task request during its duration are uniformly generated, assuming task request resource demands follow the uniform distribution. Unfortunately, the Google task request trace did not provide the information from which of its web portals each task request comes. To suit for our purpose, we randomly assign the task requests to the six web portals. In addition, we randomly generate a tolerable value β_q of resource demand violation requirements for the task requests from 1 to 3; that is, given one hour as a SLA monitoring period and each time slot is 5 minutes, each task request tolerates 5 to 15 minutes resource violations within the SLA monitoring period.

We evaluate the proposed algorithm against another algorithm that only considers the consolidation of a pair of VMs. For the sake of simplicity, we refer to the proposed algorithm and the mentioned algorithm as `Correlation` and `Two-Correlation`, respectively.

B. Performance evaluation

We first evaluate algorithm `Correlation` against algorithm `Two-Correlation` in terms of accumulated number of admitted task requests and the electricity cost of processing the task requests. Fig 3 (a) shows that algorithm `Correlation` admits more task requests than that of algorithm `Two-Correlation` in general, since algorithm

TABLE II: The maximum and minimum resource demands by different types of task requests [8], [20], [30]

Type	Duration (hrs)	CV in Duration	Cores	CV in Core
1	(0, 2]	0.3	(0, 0.2]	0.25
2	(0, 2]	0.15	(0.2, 0.5]	0.04
3	(0, 2]	0.13	(0.5, 4.0]	0.03
4	(0, 2]	0.1	(0.5, 4.0]	0.06
5	(2, 24]	0.1	(0, 0.2]	0.30
6	(2, 24]	0.09	(0, 0.2]	0.15
7	(2, 24]	0.06	(0.2, 4.0]	0.05
8	(2, 24]	0.05	(0.2, 4.0]	0.08
Type	Memory (GBs)	CV in Memory	Storage (GBs)	CV in Storage
1	(0, 0.5]	0.10	(0.03, 0.22]	0.8
2	(0, 2.0]	0.09	(0.03, 0.22]	0.8
3	(0, 1.0]	0.07	(0.09, 1.65]	0.51
4	(1, 2.0]	0.03	(0.09, 1.65]	0.51
5	(0, 0.5]	0.15	(0.11, 1.0]	0.38
6	(1, 2.0]	0.10	(0.11, 1.0]	0.38
7	(0, 1.0]	0.05	(1.65, 2.83]	0.35
8	(1, 2.0]	0.04	(1.65, 2.83]	0.35

`Correlation` allows more VMs to be consolidated into a single server, thereby creating more opportunities for resource sharing and thus improving resource utilizations. Figs 3 (b) and (c) show the electricity cost curves by the two algorithms, from which it can be seen that the electricity cost by algorithm `Correlation` is larger than that by algorithm `Two-Correlation`, since serving more task requests will incur higher electricity costs.

To make clear how VM consolidation works, we then show the performance of algorithm `Correlation` in terms of the average number of VMs that are consolidated together in servers with different platforms. Fig 4 shows the average number of VMs consolidated in a single server by algorithm `Correlation`. It can be seen that platform C allows the largest number of VMs to be consolidated in a single server, since the servers in platform C have the largest capacities in terms of CPU, memory, and storage as shown in Table I.

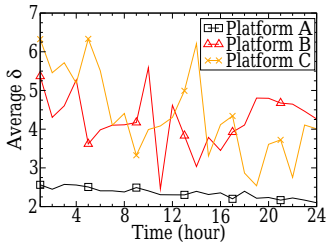
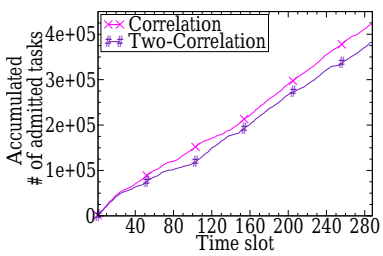
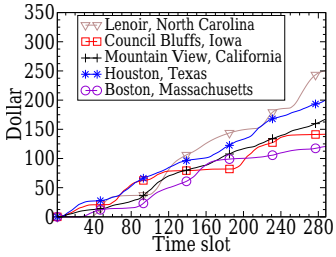


Fig. 4: Average δ of different platforms in Table I by algorithm `Correlation`

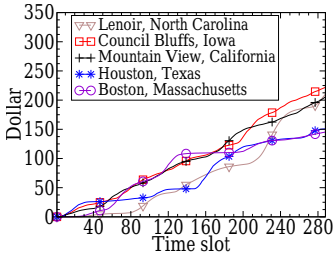
To evaluate the effectiveness of our framework, we finally show the average number of resource demand violations by different types of task requests by algorithms `Correlation` and `Two-Correlation`. Fig 5 depicts the average number of resource demand violations of different types of task requests in a SLA evaluation period by algorithms `Correlation` and `Two-Correlation`. Intuitively, the resource demand violations by algorithm `Correlation` will be higher than that of algorithm `Two-Correlation` as algorithm `Correlation` allows more VMs sharing resources and thus has high probability to violate VM demands. However, it can be seen from the figure that the solution of algorithm



(a) Accumulated number of admitted task requests



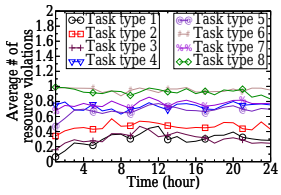
(b) Average electricity cost by algorithm `Correlation`



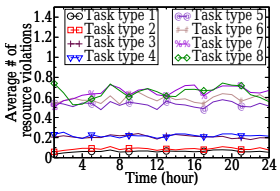
(c) Average electricity cost by algorithm `Two-Correlation`

Fig. 3: Algorithm performance

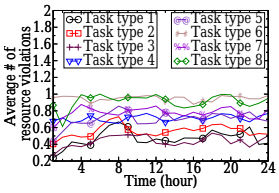
`Correlation` has no more resource demand violations than that by algorithm `Two-Correlation`. Also, it can be seen that task types from 5 to 8 have higher resource demand violations than task types from 1 to 4, because task requests of types from 5 to 8 have more resource demands than these of types from 1 to 4.



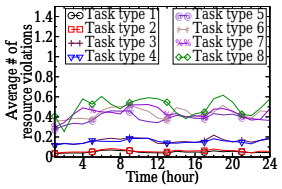
(a) Memory resource demand violations by algorithm `Correlation`



(b) Storage resource demand violations by algorithm `Correlation`



(c) Memory resource demand violations by algorithm `Two-Correlation`



(d) Storage resource demand violations by algorithm `Two-Correlation`

Fig. 5: The performance of algorithms `Correlation` and `Two-Correlation` in terms of resource demand violations during a SLA monitoring period

VI. Conclusion

In this paper, we considered the electricity cost minimization problem in a distributed cloud with heterogeneous cloud resources and user demands, by proposing an optimization framework that dispatches task requests from different web portals to different data centers, followed by resource provisioning for different types of VMs with complementary resource demands within each data center. We also evaluated the performance of the proposed algorithms through experimental simulations with real electricity prices and task request traces. Experimental results demonstrate that the proposed algorithms are promising.

References

[1] M. A. Adnan, R. Sugihara, and R. Gupta. Energy efficient geographical load balancing via dynamic deferral of workload. *Proc. of CLOUD*, IEEE, 2012.

[2] Amazon EC2. `http://aws.amazon.com/ec2`.

[3] U.S. energy information administration (EIA). `http://www.eia.doe.gov/`.

[4] Federal Energy Regulatory Commission. `http://www.ferc.gov/`.

[5] A. Gandhi, V. Gupta, M. Harchol-Balter, and A. Kozuch. Optimality analysis of energy-performance trade-off for server farm management. *J. Performance Evaluation*, Elsevier, Vol. 67, pp. 1155–1171, 2010.

[6] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya. SLA-Based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. *Proc. of ICA3PP*, LNCS, Vol. 7016, Springer, 2011.

[7] N. Garg and J. Könemann. Faster and simpler algorithms for multi-commodity flow and other fractional packing problems. *Proc of FOCS'98*, IEEE, pp.300–309, 1998.

[8] Googleclusterdata–traces of google workloads. *http://code.google.com/p/googleclusterdata/*.

[9] D. Gross and C. Harris. Fundamentals of queueing theory. `Wiley`, 1998.

[10] HP Servers. *http://www8.hp.com/au/en/products/proliant-servers*.

[11] A. Kansal, F. Zhao, N. Kothari, and A. A. Bhattacharya. Virtual machine power metering and provisioning. *Proc. of SoCC*, ACM, 2010.

[12] J. Kaplan , W. Forrest, and N. Kindler. Revolutionizing data centre energy efficiency. *McKinsey*, Technique Report, 2008.

[13] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The bell system technical journal*, Vol. 49, No. 1, pp. 291–307, 1970.

[14] J. Kim, M. Ruggiero, D. Atienza, and M. Lederberger. Correlation-aware virtual machine allocation for energy-efficient datacenters. *Proc. of DATE*, IEEE, 2013.

[15] J. Koomey. Growth in data center electricity use 2005 to 2010. *Analytics Press*, Aug. 2011.

[16] B. Lin and P. A. Dinda. VSched: mixing batch and interactive virtual machines using periodic real-time scheduling. *Proc. of SC*, IEEE, 2005.

[17] M. Lin, A. Wierman, L. Andrew, and E. Thereska. Dynamic right-sizing for power-proportional data centers. *Proc. of INFOCOM*, IEEE, 2011.

[18] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew. Geographical load balancing with renewables. *ACM SIGMETRICS Performance Evaluation Review*, Vol. 39, pp. 62–66, 2011.

[19] X. Meng, C. Isci, J. Kephart, L. Zhang, and E. Bouillet. Efficient resource provisioning in compute clouds via VM multiplexing. *Proc. of ICAC*, ACM, 2010.

[20] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das. Towards characterizing cloud backend workloads: insights from Google compute clusters. *SIGMETRICS Performance Evaluation Review*, ACM, Vol. 37, No. 4, pp. 34–41, 2010.

[21] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.

[22] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for Internet-scale systems. *Proc. of SIGCOMM*, ACM, 2009.

[23] C. Reiss, A. Tumanov, G. Ganger, R. Katz, and M. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *Proc. of SoCC*, ACM, 2012.

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

[25] T. White. *Hadoop: The Definitive Guide*. 4th edition, O'Reilly, 2015.

[26] Z. Xu and W. Liang. Minimizing the operational cost of data centers via geographical electricity price diversity. *Proc. of CLOUD*, IEEE, 2013.

[27] Z. Xu, W. Liang, and Q. Xia. Efficient virtual network embedding via exploring periodic resource demands. *Proc. of LCN*, IEEE, 2014.

[28] Z. Xu and W. Liang. Operational cost minimization for distributed data centers through exploring electricity price diversity. *Computer Networks*, Vol. 83, pp59–75, Elsevier, 2015.

[29] J. Zhan, L. Wang, X. Li, W. Shi, C. Weng, W. Zhang, and X. Zang. Cost-aware cooperative resource provisioning for heterogeneous workloads in data centers. *Trans. on Computers*, IEEE, Vol. 62, No. 11, pp. 2155–2168, 2013.

[30] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein. HARMONY: Dynamic Heterogeneity-aware resource provisioning in the cloud. *Proc. of ICDCS*, IEEE, 2013.