Dynamic Fine-Grained Resource Provisioning for Heterogeneous Applications in Virtualized Cloud Data Center

Jing Bi[1], Haitao Yuan[2], Yushun Fan[1], Wei Tan[3], and Jia Zhang[4]
[1]National Laboratory for Information Science and Technology Tsinghua University Beijing 10084, China
[2]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[3]IBM Thomas J. Watson Research Center Yorktown Heights, NY 10598, USA
[4]Carnegie Mellon University Silicon Valley
{bijing, fanyushun}@mail.tsinghua.edu.cn, yuanhaitao@buaa.edu.cn, wtan@us.ibm.com, jia.zhang@sv.cmu.edu

***Abstract*—The balance between customer-perceived application performance and cloud provider's profit is a key to achieve win-win in cloud economy. Current researches on cloud resource allocation do not sufficiently address the issue of minimizing energy cost and maximizing revenue for various applications in virtualized cloud data center (VCDC). This paper presents a new approach to realize the optimization of VCDC's profit based on the service-level agreements between cloud providers and customers. A precise model of the external and internal request arrival rates is proposed for virtual machines of different service classes. An analytic probabilistic model is then developed for non-equilibrium VCDC states. Next, a smart controller is proposed for fine-grained resource provisioning and sharing among multiple applications. A novel hybrid meta-heuristic algorithm based on simulated annealing and particle swarm optimization is developed to solve the formulated profit maximization problem. The proposed algorithm can guarantee that differentiated service qualities can be provided with higher overall performance and lower energy cost. Finally, the effectiveness of the proposed approach is validated with trace-driven simulation.**

***Keywords*-Data center; dynamic resource provisioning; heuristic algorithm; optimization**

I. Introduction

With the wide deployment of cloud computing services, virtualized cloud data centers (VCDCs) become more and more important. Many kinds of intensive applications, such as CPU and I/O intensive ones, concurrently run in VCDC and require various infrastructure resources [1]. Traditional resource allocation for a single intensive application is inefficient, since it can lead to much waste of resources. For example, CPU-intensive applications may occupy CPU resources for a long time, but cause a waste of disk I/O resources in a physical machine (PM) or virtual machine (VM). Moreover, due to the increasing energy cost of VCDCs [2], it is inappropriate to increase the number of servers in VCDCs at its current pace. It is thus challenging for VCDC administrators to meet a service level agreement (SLA) due to the dynamic multi-resource sharing among various intensive applications.

Several dynamic resource provisioning methods have been proposed to effectively allocate resources for intensive applications. Unfortunately, most existing methods fail to realize the objectives to minimize provider's energy cost and to maximize revenue in complex cloud environments. They mainly focus on a single type of resources even in multi-resource environments where customers have heterogeneous resource requirements [3]. However, requests may have varying demands for CPU, memory, and I/O resources. In addition, the virtualization technology consolidates multiple online application services into fewer physical resources. These resource provisioning strategies offer dynamic VM provisioning, workload consolidation and efficient operation of VMs and PMs. They are very helpful for VCDC to achieve high utilization and energy efficiency, and can greatly improve the traditional off-line capacity planning process [4]. Note that, the static provisioning is usually considered as a benchmark to evaluate new methods.

This study provides a way to allocate various heterogeneous resources to requests of every intensive application in VCDC. It enables *dynamic fine-grained resource provisioning*, which turns on a minimum number of VMs to meet the current demand and dispatches the workload among running VMs to meet SLAs. In this regard, it is significant to realize the high overall utilization of infrastructure resources, minimization of energy cost, and maximization of the revenue of a VCDC provider. Therefore, we focus on *minimizing energy expenditure* and *maximizing revenue* while meeting the demand of various intensive application services.

The key contributions of this paper are two-fold:

- Firstly, we propose a novel *Smart Controller* (SC) to support a dynamic fine-grained resource provisioning method in a *non-equilibrium states* VCDC.
- Secondly, we propose a *hybrid meta-heuristic* algorithm to determine the allocation of CPU and I/O resources, respectively, where the revenue of application services is maximized and machine-level energy expenditure is minimized.

The rest of the paper is organized as follows. Section II discusses the related work. Section III describes the motivation and VCDC architecture. Section IV constructs a system model. Section V formulates the profit maximization problem of

2159-6190/15 $31.00 © 2015 IEEE
DOI 10.1109/CLOUD.2015.64

multiple resources, and proposes an algorithm to solve it. Section VI presents its performance evaluation results. Section VII concludes this paper.

II. Related work

Recently, a few studies have examined SLA resource allocation issues for data centers, but they cannot be readily adapted to cloud computing environments because they usually assume equilibrium states and adopt mean value analysis. For example, Urgaonkar et al. present an analytical model of dynamic resource provisioning for multi-tier clusters [5]. However, they assume that available resources are always sufficient, and fail to consider total profit maximization based on different performance demands. Lama et al. propose an efficient resource allocation optimization model [6]. Its integration with an independent fuzzy controller provides superior performance in resource utilization and end-to-end response time guarantee. However, the resource contention problem is not addressed in their work. They fail to provide heterogeneous server configuration in virtualized systems. Goudarzi et al. pose an SLA-based resource allocation problem for cloud computing environments [7]. They consider CPU, memory and network resource requirement. However, their single and simple M/M/1 queueing system cannot reflect real cases well.

Furthermore, a virtualization technology is an efficient resource sharing approach to support various intensive applications in VCDC [8]. It can allocate physical resources to separate VMs and realize application isolation. However, the high variability of workload poses a challenge to accurately predict the requirement of each resource. For example, Kalyvianaki et al. adopt Kalman filters to track and control CPU utilization in virtualized environments [9]. Khazaei et al. present a fine-grained performance model for homogeneous resources assignment in cloud computing centers [10]. In contrast to these researches, our work can support a fine-grained and heterogeneous resource allocation for a virtualized cloud computing environment. Padala et al. present a resource control system that achieves application performances by automatically adapting to dynamic workload changes [11]. They provide a novel multi-input, multi-output (MIMO) resource controller to manage multiple resources. However, our method can not only provide differentiated service qualities but also reduce energy cost. Moreover, we can accurately compute request arrival rates according to various intensive applications of different service classes in a VCDC.

Different from previous works, based on the variability of workload for various applications, this work provides dynamic fine-grained allocation for each virtualized resource by a model that considers non-steady state situations through the probabilistic analysis of a VCDC performance.

III. System Architecture

One particular motivation of this work is due to the clear need to pack together applications with complementary multi-resource allocation requirements, such as placing a CPU-intensive and an I/O-intensive VM on the same PM. We propose a *Smart Controller* (SC) architecture for this purpose, as shown in Fig. 1.

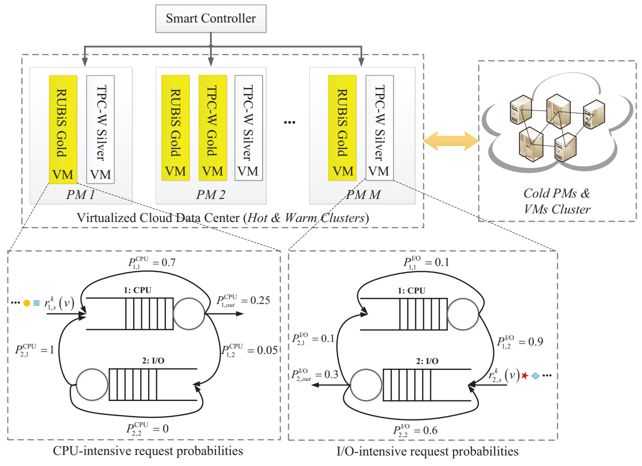


Fig. 1. System architecture.

To reduce overall VCDC provisioning and energy cost, and meet the SLA of Gold and Silver services for various applications, we assume that PMs and VMs are categorized into three states: hot (i.e., turned on with running), warm (turned on, but not ready), and cold (turned off) [12]. Turned on PMs and VMs are placed in hot and warm clusters, while turned off ones are placed in cold cluster to reduce energy consumption during periods of small workload. Our proposed SC can smartly allocate internal and external workloads to designated VMs corresponding to different applications. SC can also minimize the consumption of computing and storage resources by specifying the number of PMs and VMs, as well as CPU and I/O shares per VM. Due to different performance levels of various service applications, in our work, Gold services pose higher performance requirement than silver ones. Besides, the operating systems are encapsulated into each seperated VM. Thus, multiple VMs can be used for parallel processing of different intensive applications.

IV. System Model

A. System Dynamics

A VCDC is a group of VMs distributed across one or more PMs, cooperating to host multiple applications. Its dynamics for Gold and Silver service classes at multiple resources is described by a discrete-time state space equation:

$$\eta_s^k(\tau+1)=\psi\left(\eta_s^k(\tau),\sigma_s^k(\tau),\lambda_s^k(\tau)\right) \tag{1}$$

where $\eta_s^k(\tau)$ is a system state at time $\tau\in\{1,2,...,t\}$. $\sigma_s^k(\tau)$ denotes control variables, and $\lambda_s^k(\tau)$ is a system input at time τ. Function ψ captures the relationship among states, control variables, and system inputs.

Its state of the s-th service class is denoted as:

$$\eta_s^k(\tau)=\left(E(T_s^k(\tau)),q_{i,s}^k(\tau),\gamma_s^{k,\mathbb{C}}(\tau),\gamma_s^{k,\mathbb{N}}(\tau)\right) \tag{2}$$

where $E(T_s^k(\tau))$ is its expected average response time. $q_{i,s}^k(\tau)$ is the expected number of queued requests into the *i*-th resource type. $\gamma_s^{k,\mathbb{C}}(\tau)$ and $\gamma_s^{k,\mathbb{N}}(\tau)$ are the numbers of finished and rejected requests in SLA of the *k*-th intensive application for the *s*-th service class, respectively.

Its control variables related to the *s*-th service class are denoted as:

$$\sigma_s^k(\tau)=\left(N(\tau),V(\tau),\varphi_s^k(\tau),c_s^k(\tau),\phi_{s,v}^k(\tau),\theta_{s,v}^k(\tau),\omega_{s,v}^k(\tau)\right) \tag{3}$$

where $N(\tau)$ and $V(\tau)$ are the number of turned on PMs and VMs, respectively. $\varphi_s^k(\tau)$ and $c_s^k(\tau)$ are system-wide control variables indicating the number of hot PMs and VMs, respectively. $\phi_{s,v}^k(\tau)$ and $\theta_{s,v}^k(\tau)$ are CPU and I/O allocations of the *v*-th VM in the *k*-th intensive application for the *s*-th service class, respectively. $\omega_{s,v}^k(\tau)$ is workload fraction directed to the *v*-th VM. Note that for the sake of simplicity, throughout the remainder of this paper, we will simplify notations that contain τ, and remove τ from these notations. For example, $c_s^k(\tau)$ is simplified as c_s^k.

The system input λ_s^k is the workload arrival rate. We design ψ as a difference model for a VCDC. The average arrival rate of the *k*-th intensive application for the *s*-th service class into the *i*-th resource type in a VM is given by:

$$\lambda_{i,s}^k=r_{i,s}^k+\sum_{j=1}^{I}\lambda_{j,s}^k\cdot P_{(j,s),(i,s)}^k \tag{4}$$

where $r_{i,s}^k$ is an external workload arrival rate of a Poisson process, and we adopt the Gauss-Seidel iterative method [13] to approximate $\lambda_{i,s}^k$ and $\lambda_{j,s}^k$.

Then, the total average arrival rate from both the internal and external of all intensive applications for the *s*-th service class into the *i*-th resource type in a PM is denoted by $\Lambda_{i,s,n}$, i.e.,

$$\Lambda_{i,s,n}=\sum_{k=1}^{K}\lambda_{i,s}^k=\sum_{k=1}^{K}\left[r_{i,s}^k+\sum_{j=1}^{I}\lambda_{j,s}^k\cdot P_{(j,s),(i,s)}^k\right] \tag{5}$$

where $\lambda_{i,s}^k$ ($\lambda_{j,s}^k$) and $r_{i,s}^k$ denote average and external arrival rate of the request of the *k*-th intensive applications for the *s*-th service class into resource type *i* (*j*), respectively. $P_{(j,s),(i,s)}^k$ denotes the probability that when a *k*-th intensive application finishes at resource type *j*, it next moves to resource type *i* for the *s*-th service class.

Thus, the total average arrival rate from both the internal and external of all intensive applications for the *s*-th service class into the *i*-th resource type is given by:

$$\Lambda_{i,s}=\sum_{n=1}^{\varphi}\Lambda_{i,s,n} \tag{6}$$

The service rate $\mu_{i,s}^k$ of a VCDC is determined by the number of VMs, and CPU and disk I/O allocations given to each VM. Each VM is assigned a share of the PM's CPU, memory, and disk I/O. In addition, each VM uses the function $f(\cdot)$ to map CPU $\phi_{s,v}^k$ and disk I/O $\theta_{s,v}^k$ allocations of the VM *v* in a VCDC to a corresponding processing rate. Therefore, we can obtain the following equations:

$$\mu_{i,s}^k=\sum_{k=1}^{K}\mu_{i,s}^k=\sum_{k=1}^{K}\sum_{v=1}^{c_s^k}\mu_{i,s,v}^k \tag{7}$$

$$\left[\mu_{1,s,v}^k,\ldots,\mu_{I,s,v}^k\right]=f(\phi_{s,v}^k,\theta_{s,v}^k). \tag{8}$$

We assume that the sampling period is *T*, e.g. 30 seconds, to capture the system dynamics. The initially measured queueing length is $q_{i,s}^k(\tau-1)$ at this time period. When $\lambda_{i,s}^k$ is small, a VCDC is not fully utilized. At this time period, we first measure the whole real service rate $\mu_{i,s}^k$ of a VCDC. We assume that request arrival rate $\lambda_{i,s}^k$ and service rate $\mu_{i,s}^k$ are fixed over each time interval of length *T*. The instantaneous queueing length $q_{i,s}^k(\tau)\geq 0$ at any time τ in the next sampling period is obtained by using the current queueing length $q_{i,s}^k(\tau-1)$, incoming workload $\lambda_{i,s}^k$ dispatched to a VCDC, and service rate $\lambda_{i,s}^k$. It can be obtained by:

$$q_{i,s}^k(\tau)=\max\left\{q_{i,s}^k(\tau-1)+(\lambda_{i,s}^k-\mu_{i,s}^k)\cdot T,0\right\}. \tag{9}$$

That is, the queueing length $q_{i,s}^k(\tau)$ at any time τ in the next sampling period equals to the current queueing length $q_{i,s}^k(\tau-1)$ plus new arrivals of service requests, and minus the number of service requests that are handled by a VCDC within sampling period *T*.

Based on (9), the average length of the queue in the next sampling period is:

$$\bar{q}_{i,s}=\sum_{k=1}^{K}\bar{q}_{i,s}^k=\frac{1}{t}\sum_{k=1}^{K}\sum_{\tau=1}^{t}q_{i,s}^k(\tau). \tag{10}$$

In order to calculate the average response time of requests in the next sampling period, we consider the following two cases based on arrival and service rates.

1) If $\lambda_{i,s}^k<\mu_{i,s}^k=\sum_{v=1}^{c_s^k}\mu_{i,s,v}^k$, the system is underloaded, i.e., the *i*-th type resource is enough to process all requests of the *k*-th intensive application for the *s*-th service class in a VCDC. Therefore, the system can stay at its steady state. Here, the whole actual service rate $\mu_{i,s}^k$ is related to $\lambda_{i,s}^k$. Besides, the queueing length in the next sampling period decreases with time. Based on the initial value of current queueing length $q_{i,s}^k(\tau-1)$, we further consider the two sub-cases: a) if $q_{i,s}^k(\tau-1)=\bar{q}_{i,s}^k(t)$, we assume that the queueing system has already entered a relatively steady state. Therefore, the response time $E(T_{i,s})$ is calculated via a steady model [8]; b) if $q_{i,s}^k(\tau-1)>\bar{q}_{i,s}^k(t)$, the queue length begins to decrease from the initial length $q_{i,s}^k(\tau-1)$. Then $\bar{q}_{i,s}(t)$ can be rewritten as:

$$\bar{q}_{i,s}=\sum_{k=1}^{K}\bar{q}_{i,s}^k=\frac{1}{t}\sum_{k=1}^{K}\{[q_{i,s}^k(\tau-1)+\frac{\hat{T}}{2}\cdot(\lambda_{i,s}^k-\mu_{i,s}^k)]\cdot\hat{T}$$
$$+L_q\cdot(t-\hat{T})\} \tag{11}$$

where L_q denotes the average number of requests waiting in the queue. Based on the preceding steady state analytic model, it is calculated as:

$$L_q = \sum_{m=\sum_{k=1}^{K} c_s^k}^{\infty} p_m \cdot (m - \sum_{k=1}^{K} c_s^k).$$

where p_m is the probability of the case that there are *m* requests in the queue. We adopt the birth and death state equilibrium equations of Markov processes [14] to obtain p_m.

Then, the average response time is obtained as:

$$E(T_{i,s}) = \frac{\bar{q}_{i,s} + L_a}{\Lambda_{i,s}} \tag{12}$$

where L_a denotes the average number of requests that are being processed in the queue. Based on the preceding steady state analytic model, we have:

$$L_a = \sum_{m=1}^{\sum_{k=1}^{K} c_s^k - 1} p_m \cdot m + \sum_{m=\sum_{k=1}^{K} c_s^k}^{\infty} p_m \cdot \sum_{k=1}^{K} c_s^k.$$

2) If $\lambda_{i,s}^k \geq \mu_{i,s}^k = \sum_{v=1}^{c_s^k} \mu_{i,s,v}^k$, the system is at an overloaded state, i.e., it cannot stay steady. Therefore, the whole actual service rate is $\mu_{i,s}^k$. The queueing length in the next sampling period increases, i.e.,

$$\bar{q}_{i,s} = \sum_{k=1}^{K} \bar{q}_{i,s}^k = \sum_{k=1}^{K} \{ q_{i,s}^k(\tau - 1) + \frac{t}{2} \cdot (\lambda_{i,s}^k - \mu_{i,s}^k) \}. \tag{13}$$

Then, given the average queueing length, based on the Little's Law [15], we have the average response time:

$$E(T_{i,s}) = \frac{\bar{q}_{i,s} + \sum_{k=1}^{K} c_{i,s}^k}{\mu_{i,s}}. \tag{14}$$

Let $E(T_{i,s})$ denote the expected average response time in a VCDC of the *k*-th intensive application for the *s*-th service class.We use subscript $\tilde{k}$ to show resource type *k*, and superscript *k* the *k*-th intensive application.

$$E(T_s^k) = P_{\tilde{k},out}^k \cdot E(T_{\tilde{k},s}) + P_{\tilde{k},\tilde{k}}^k \cdot [E(T_{\tilde{k},s}) + E(T_s^k)] + \sum_{i=1, i \neq \tilde{k}}^{I} P_{\tilde{k},i}^k \cdot P_{i,\tilde{k}}^k \cdot [E(T_{\tilde{k},s}) + E(T_{i,s}) + E(T_s^k)] \tag{15}$$

where $E(T_{\tilde{k},s})$ ($E(T_{i,s})$) denotes expected average response time of resource type $\tilde{k}$ (*i*) for the *s*-th service class. $P_{\tilde{k},out}^k$ means the probability that the *k*-th intensive application from type resource $\tilde{k}$ leaves VM *v*. $P_{\tilde{k},\tilde{k}}^k$ expresses the probability that it from resource type $\tilde{k}$ returns to resource type $\tilde{k}$ to repeat the process. $P_{\tilde{k},i}^k$ shows the probability that it from resource type $\tilde{k}$ goes to resource type *i*. $P_{i,\tilde{k}}^k$ denotes the probability that the *k*-th intensive application from resource type *i* returns to resource type $\tilde{k}$ to repeat the process. We assume that the time for deploying an application, turning on and migration VM is ignorable.

B. Energy Consumption

In order to reduce VCDC's machine-level energy consumption, the number of hot servers should be dynamically adjusted according to the rate of receiving service requests. Each server can only serve one request at a time in VCDC. However, a VCDC typically runs multiple VMs on each PM. It is thus highly desired to pack together requests with complementary resource requirements. We thus use CPU utilization as the main signal of machine-level activity. Therefore, we model energy consumption for a VCDC such that it is proportional, roughly linear, to its utilization. Multiple studies have shown that CPU utilization is indeed a good estimator for power usage [16]. We use V_{max} to denote the maximal number of VMs in a VCDC. Let $c \leq V$ denote that the number of hot VMs c is not more than that of turn on VMs V at time τ. We then have the machine-level power usage of a VCDC:

$$\mathbb{E}(u) = F(V) + A(u, V) + \epsilon \tag{16}$$

where $u \in [0, 1]$ denotes its average CPU utilization of the *k*-th intensive application for the *s*-th service class at time τ. *F*, *A*, and ϵ are the fixed power, variable power, and empirically derived correction constant, respectively [17]. Note that in our works, ϵ is set to zero.

$$F(V) = V \cdot (\mathbb{E}_{idle} + (U - 1) \cdot \mathbb{E}_{peak}) \tag{17}$$

$$A(u, V) = V \cdot (\mathbb{E}_{peak} - \mathbb{E}_{idle}) \cdot u \tag{18}$$

where $\mathbb{E}_{peak}$ denotes the average peak power when a VM is handling a service request. $\mathbb{E}_{idle}$ is the average idle power draw of a single VM of the *k*-th intensive application for the *s*-th service class. U is the power usage effectiveness of a VCDC. From (16), the energy consumption at a VCDC increases as we turn on more VMs or hot VMs at higher utilization.

V. Profit maximization problem

A. Optimization Problem Formulation

In order to maximize profit, this work presents a profit function. We focus on the multiple resource allocation problem for various intensive applications of different service classes in a VCDC. If $\eta(\tau)$ denotes the operating state and $\sigma(\tau)$ is control variable, the profit generated at time τ is given by:

$$Profit(\eta(\tau), \sigma(\tau)) = Revenue - Cost \tag{19}$$

where revenue is determined by whether SLA is met or not from the corresponding function, if $E(T_s^k) \leq \overline{T}_s^k$, "meeting SLA" is of a reward type; otherwise, "violating SLA" is of a refund type or loss one. Cost is the machine-level energy consumption as incurred by hot PMs and VMs according to their operational states.

1) Revenue Modeling

The total revenue function collected by a VCDC at sampling period T can be calculated as:

$$Revenue=\begin{cases}\sum\limits_{k=1}^{K}\sum\limits_{s=1}^{S}\left[\mathbb{C}_s^k\cdot\gamma_s^{k,\mathbb{C}}\right]\cdot T & if\ E(T_s^k)\leq\overline{T}_s^k\\ \sum\limits_{k=1}^{K}\sum\limits_{s=1}^{S}\left[\mathbb{C}_s^k\cdot\gamma_s^{k,\mathbb{C}}-\mathbb{N}_s^k\cdot\gamma_s^{k,\mathbb{N}}\right]\cdot T & otherwise\end{cases} \tag{20}$$

where $\mathbb{C}_s^k$ and $\mathbb{N}_s^k$ are the unit request revenue and refund of the k-th intensive application for the s-th service class, respectively. According to the predicted result of the system metrics, we can conclude that if $\sum\limits_{i=1}^{I}\lambda_{i,s}^k<\sum\limits_{i=1}^{I}\mu_{i,s}^k$, $\gamma_s^{k,\mathbb{C}}=\sum\limits_{i=1}^{I}\lambda_{i,s}^k\cdot P(E(T_s^k)\leq\overline{T}_s^k)\cdot t$ and $\gamma_s^{k,\mathbb{N}}=0$. $\gamma_s^{k,\mathbb{C}}$ and $\gamma_s^{k,\mathbb{N}}$ denote the number of finished and rejected requests for the k-th intensive application of the s-th service class, respectively. Otherwise, if $\sum\limits_{i=1}^{I}\lambda_{i,s}^k\geq\sum\limits_{i=1}^{I}\mu_{i,s}^k$, to take advantage of the steady-state queueing network model, we use a binary search method to determine the threshold of the request arrival rates, denoted as $\sum\limits_{i=1}^{I}\Lambda_{i,s}^{k*}$. Therefore, the number of finished and rejected requests for the k-th intensive application of the s-th service class is $\gamma_s^{k,\mathbb{C}}=\sum\limits_{i=1}^{I}\Lambda_{i,s}^{k*}\cdot P(E(T_s^k)\leq\overline{T}_s^k)\cdot t$ and $\gamma_s^{k,\mathbb{N}}=\left[\sum\limits_{i=1}^{I}\lambda_{i,s}^k-\sum\limits_{i=1}^{I}\Lambda_{i,s}^{k*}\cdot P(E(T_s^k)\leq\overline{T}_s^k)\right]\cdot t$, respectively. $(\mathbb{C}_s^k\cdot\gamma_s^{k,\mathbb{C}})\cdot T$ denotes the total revenue received by a VCDC within time interval T for the requests of the k-th intensive application for the s-th service class that are handled before an SLA-deadline. $(\mathbb{N}_s^k\cdot\gamma_s^{k,\mathbb{N}})\cdot T$ denotes the total refund paid to customers within time interval T for the requests of the k-th intensive application for the s-th service class that are not handled before an SLA-deadline.

2) Cost Modeling

The machine-level power-consumption cost of a VCDC is usually determined by unit-time power usage $\mathbb{E}(u)$, that is, the total energy cost of hot and warm VMs. The request arrival rate of a VCDC is $P(E(T_s^k)\leq\overline{T}_s^k)\cdot\sum\limits_{i=1}^{I}\lambda_{i,s}^k$ or $P(E(T_s^k)\leq\overline{T}_s^k)\cdot\sum\limits_{i=1}^{I}\Lambda_{i,s}^{k*}$ service requests per second. A VCDC's average CPU utilization at time τ can be obtained as:

1. unloaded:

$$u=\sum_{k=1}^{K}\sum_{s=1}^{S}\left[\frac{\lambda_{i,s}^k\cdot P(E(T_s^k)\leq\overline{T}_s^k)}{\mu_{i,s}^k}\right] \tag{21}$$

2. overloaded:

$$u=\sum_{k=1}^{K}\sum_{s=1}^{S}\left[\frac{\Lambda_{i,s}^{k*}\cdot P(E(T_s^k)\leq\overline{T}_s^k)}{\mu_{i,s}^k}\right] \tag{22}$$

where since we consider the processing capacity of CPU only, resource type i refers to CPU. Then, the machine-level energy consumption associated with a VCDC at time τ can be given by:

$$\mathbb{E}(u)=V\cdot\left[(\mathbb{E}_{idle}+(U-1)\cdot\mathbb{E}_{peak})+(\mathbb{E}_{peak}-\mathbb{E}_{idle})\cdot u\right] \tag{23}$$

Let χ denote the instantaneous electricity price. Therefore, the total machine-level energy consumption cost at the sampling period T can be calculated as:

$$Cost=T\cdot\chi\cdot\mathbb{E}(u) \tag{24}$$

In Section IV, we will use the pricing information to obtain VCDC's cost of electricity.

3) Profit Maximization

In this paper, we assume that a workload admission control policy to a VCDC is provided ahead of time. The resource allocation problem in question is how to dynamically allocate CPU and I/O resources among VMs with the goal of maximizing the global profit function, i.e., our work focuses on minimizing VCDC's energy expenditure and maximizing their revenue for various intensive application services. Our proposed SC's goal is to find optimal CPU and I/O resource allocations of N PMs for the set of V VMs.

Given the profit function in (19), we formulate a VCDC resource provisioning problem as follows:

$$f_1=\max_{\eta,\sigma}\sum_{y=\tau+1}^{\tau+h}Revenue(\eta(y),\sigma(y))$$

$$f_2=\min_{\eta,\sigma}\sum_{y=\tau+1}^{\tau+h}Cost(\eta(y),\sigma(y))$$

s.t.

$$\sum_{k=1}^{K}\sum_{s=1}^{S}\varphi_s^k(y)\leq N(y)\leq N_{max},\quad \forall s=1,...,S,k=1,...,K \tag{25}$$

$$\sum_{k=1}^{K}\sum_{s=1}^{S}c_s^k(y)\leq V(y)\leq V_{max} \tag{26}$$

$$c_s^k(y)\geq\mathbb{Q}_{min} \tag{27}$$

$$\sum_{v=1}^{c_s^k(y)}\omega_{s,v}^k(y)=1 \tag{28}$$

$$\sum_{k=1}^{K}\sum_{s=1}^{S}\sum_{v=1}^{c_{s,n}^k(y)}d_{s,v,n}^k(y)\cdot\phi_{s,v}^k(y)\leq H_{max}^n,\quad d_{s,v,n}^k(y)\in\{0,1\},n\in\{1,...,N_{max}\} \tag{29}$$

$$\sum_{k=1}^{K}\sum_{s=1}^{S}\sum_{v=1}^{c_{s,n}^k(y)}d_{s,v,n}^k(y)\cdot\theta_{s,v}^k(y)\leq H_{max}^n \tag{30}$$

$$E(T_s^k(y))\leq\overline{T}_s^k \tag{31}$$

$$\begin{cases}\lambda_{i,s}^k(y)<\mu_{i,s}^k(y)\\ \lambda_{i,s}^k(y)\geq\mu_{i,s}^k(y)\end{cases} \tag{32}$$

where h denotes control interval length. The control constraints can be updated periodically at the beginning of each control interval, $h{\cdot}T$, i.e., h=5, $h{\cdot}T$=150 seconds, and are unchanged in the control interval. Constraints (25) and (26) ensure that the total number of hot PMs and VMs cannot exceed their respective maximum number at sampling period *T*. Constraint (27) forces the controller to conservatively operate at least $\mathbb{Q}_{min}$ VMs at all times to accommodate a sudden spike in request arrivals. Here, we set $\mathbb{Q}_{min} = 1$. Constraint (28) shows that $\omega^k_{s,v}(y)$ is workload fraction directed to the *v*-th VM. The decision variable $d^k_{s,v,n}(y){\in}\{0,1\}$ indicates whether the *v*-th VM of the *k*-th intensive application for the *s*-th service class is allocated to PM $n{\in}\{1,...,N_{max}\}$. Constraints (29) and (30) ensure that the cumulative CPU and I/O given to VMs does not exceed the maximum capacity available on PM *n*. Constraint (31) states that the expected average response time $E(T^k_s(y))$ cannot exceed the target response time $\overline{T}^k_s$ of the *k*-th intensive application for the *s*-th service class specified in SLA. Constraint (32) shows that the request arrival rate of the *k*-th intensive application for the *s*-th service class cannot exceed the capacity of all VMs.

B. Solution Algorithm

In the problem, the objective functions f_1 and f_2 are nonlinear. Besides, N, V, φ^k_s and c^k_s are integer variables, while $\phi^k_{s,v}$, $\theta^k_{s,v}$ and $\omega^k_{s,v}$ are continuous ones. Therefore, the problem is a mixed integer non-linear programming (MINLP) that is NP-complete. Existing methods (e.g., equality relaxation, and branch and bound) usually rely on the problem structure. Besides, they converge to global optima at the cost of long execution time. Metaheuristic algorithms are robust and do not rely on any specific structure of the problem. Though they do not guarantee to obtain global optima, they have been widely adopted to solve MINLP.

Different metaheuristic algorithms own their respective strengths and weaknesses. For example, particle swarm optimization (PSO) can converge quickly, but it may easily trap into local optima [18]. Simulated annealing (SA) can converge to global optima by accepting worse solutions based on the Metropolis criterion [19], but its convergence speed is unsatisfying. Therefore, we adopt a hybrid meta-heuristic algorithm that combines strengths of PSO and SA algorithms. In this algorithm, old and new positions of each particle are compared in terms of the achieved profit. Better solutions are directly accepted while worse ones are accepted based on the metropolis criterion of SA. Therefore, this algorithm can increase the possibility of obtaining global optima of the formulated problem. Due to the length limit, details of the hybrid meta-heuristic algorithm is omitted from this paper.

VI. Performance evaluation

A. Simulation Setup

We use two applications about different web-based service classes in our experiments including RUBiS [20], an online auction site benchmark, and TPC-W [21], a transactional web e-Commerce benchmark. To simulate the total workload, we adopt two request traces with different service classes for VMs: (1) the publicly available log files from the Soccer World Cup 1998 Web site from June 14 to July 29, 1998 [22] as the service request trend for two service classes of RUBiS, respectively; (2) the web transaction workload traces from Google's data center [23] for two service classes of TPC-W. Note that RUBiS and TPC-W are CPU-intensive and I/O-intensive applications, respectively.

Consider a VCDC with N_{max} = 500 PMs and V_{max} = 1500 VMs, respectively. The exact number of hot PMs and VMs are updated periodically at the beginning of each control interval, $h{\cdot}T$=150 seconds, respectively. For each hot VM, with power-management, the idle power consumption can be as low as 50-65% of the peak power consumption, which can range from 100-250 Watts. We assume that the energy model is 65% idle, 1.3 *U* [16]. The electricity price information is based on the real-time pricing tariffs in Illinois Zone I, on Jan. 1, 2015 [24].

In order to evaluate the applicability of the proposed model in complex cases, we set different values for parameters, as shown in Tables I, II and III.

TABLE I
ENERGY MODEL PARAMETERS FOR GOLD AND SILVER VMS.

Bounds	**Classes**	$\mathbb{E}_{peak}$ (Watts)	$\mathbb{E}_{idle}$ (Watts)	U
CPU-intensive VM	1 (Gold)	250	125	1.7
	2 (Silver)	240	120	1.5
I/O-intensive VM	1 (Gold)	130	85	1.3
	2 (Silver)	100	50	1.0

B. Analysis and Results

For comparison, we adopt two alternative resource provisioning solutions including *non-capped* [25] and *static* [26] to evaluate the proposed method in a VCDC using trace-driven simulations. Experiments with the same parameter setting are repeated. They are based on a discrete-event simulator and resources are allocated periodically. The *non-capped* method permits VMs to make the most of idle CPU and I/O resources beyond their shares. In the *static* method, the CPU and I/O shares are predefined before initiating the execution, and they remain the same during the processing. Our method can provide performance separation for multiple intensive applications in a VCDC, where available CPU and I/O resources for a VM must be part of its resident PM, while idle CPU and I/O resources are not available in a control interval of SC.

In our experiments, we first accurately compute request arrival rates based on external and internal workload for RUBiS and TPC-W applications, respectively. Furthermore, we establish an analytic probabilistic system model to deal with non-equilibrium states in a VCDC. At the same time, we apply the proposed SC to validate our fine-grained resource provisioning method.

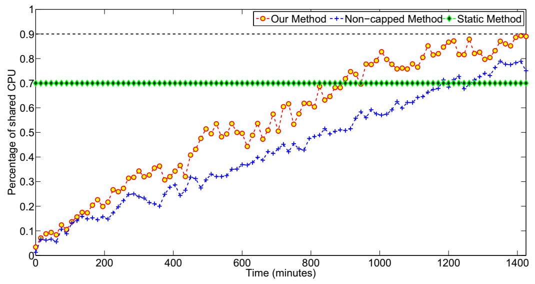
We show the results of resource usages for two intensive applications in Fig. 2. We observe that the percentages of shared CPU and I/O generated from our proposed model are higher than those of the other two methods. These results demonstrate that the system model we presented in Section

TABLE II
SLA PARAMETERS FOR RUBIS AND TPC-W.

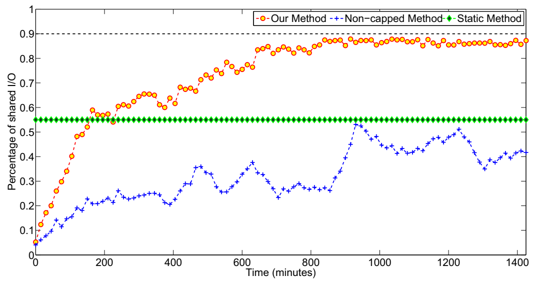
Bounds	Classes	SLA-deadline (ms)	$\mathbb{C}$ (\$ per ms)	$\mathbb{N}$ (\$ per ms)
RUBiS	1 (Gold)	300	0.00006	0.000022
	2 (Silver)	100	0.00004	0.000005
TPC-W	1 (Gold)	400	0.00007	0.000032
	2 (Silver)	200	0.00005	0.000013

TABLE III
PARAMETERS OF CPU AND I/O-INTENSIVE VM INSTANCE TYPES.

Bounds	Classes	Name	vCPU	CPU Speed (GHz)	Disk (GB)
CPU-intensive VM	1 (Gold)	c3.2Xlarge	10	2.4GHz	2 × 80 SSD
	2 (Silver)	c3.Xlarge	5	2.4GHz	2 × 40 SSD
I/O-intensive VM	1 (Gold)	i3.2Xlarge	10	2.4GHz	2 × 800 SSD
	2 (Silver)	i3.Xlarge	5	2.4GHz	1 × 800 SSD



(a)



(b)

Fig. 2. Resource allocations comparisons on a PM: (a) CPU allocations (b) Disk I/O allocations.

IV is effective to satisfy CPU and I/O resource requirements. As we can see, the *static* method fixes CPU and I/O optimal assignments to be 70% and 55% for two intensive applications, respectively. However, such resource assignment results in under-utilization or over-utilization during the entire execution period. Besides, the *non-capped* method is not suitable for I/O-intensive applications that may share the same CPU and I/O resources. Thus, the percentage of shared I/O of this method is less.

In comparison, our method first calculates accurate CPU and I/O request arrival rates before actually changing the current resources allocations for CPU and I/O intensive applications. By doing this, we are able to improve the total utilization of a VCDC and reduce the usage CPU and I/O resources. The total utilization is an important factor in saving energy cost. Reducing the frequency of resource reallocations is also important for stability of a VCDC. To avoid incidental instability of performance when VCDC resources are occupied almost completely, we both set the upper limits of CPU and I/O utilizations of each PM in SC to 90%. If the whole VCDC is overloaded, our method rejects some service requests to maximize the profit. We set control interval length in SC to $h{=}5$ and each control interval to $h{\cdot}T{=}150$ seconds. To reduce the frequency of resource reallocations, the control constraints remain unchanged in the control interval.

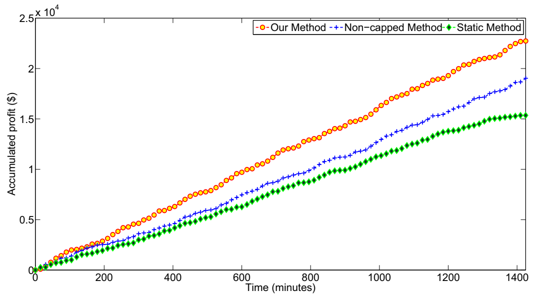


Fig. 3. Accumulated profit.

Our proposed hybrid meta-heuristic algorithm is able to maximize the revenue and minimize the energy cost, while meeting all the relevant control constraints. The result illustrated in Fig. 3 shows the accumulated total profit in a VCDC. It can be clearly shown that our method can always perform better than the *non-capped* and *static* methods. The *non-capped* method gives an ideal value for maximizing revenue. Since each resource request is always satisfied, the maximum revenue is achieved. We can obtain that although the revenue achieved by the *non-capped* method is about 12.4% higher than that achieved by our proposed method, its energy cost is about 65% higher than that of our proposed one. The reasons for the high energy cost of the *non-capped* method include: First, reallocating CPU and I/O resources are expensive, especially when it is done in each sampling period of parameter adaptation. Second, the *non-capped* method ignores service classes and request execution time. Hence, high service class request with long execution time can occupy more resources for processing. Compared with *non-capped* and *static* methods, the number of occupied resources in our model is less. Hence, our proposed method causes much less energy cost in the control interval.

Above experiments show that we can evaluate the accuracy of our resource models in comparison to the *non-capped* method and *static* baseline method. Therefore, our method can realize lower energy cost and achieve higher application profit for various application services in a VCDC.

VII. Conclusions and Future Works

This paper presents a novel analytical model to calculate profit in a virtualized cloud data center (VCDC). It considers several factors including the practical service-level agreements that currently exist between cloud providers and their customers, the amount of finished requests, the amount of rejected requests, and th electricity price. We first accurately compute request arrival rates based on the external and internal workloads for VCDC, and establish an analytic probabilistic system model to deal with non-equilibrium states in VCDC for the first time. Then, we propose a novel smart controller that can realize dynamic fine-grained resource provisioning and sharing for multiple intensive applications with different service classes. We show that the formulated optimization problem is a mixed integer non-linear programming. Then, it is solved by the proposed hybrid meta-heuristic algorithm based on particle swarm optimization and simulated annealing. Finally, simulation results based on various realistic workload traces demonstrate the accuracy of the proposed model and effectiveness of the proposed profit maximization method.

In future research, we would like to investigate how the current approach can be generalized to support different intensive application services deployed in geographically distributed cloud data centers.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (No.61033005 and No. 61174169), the National Science and Technology Support Program of China (2012BAF15G01), the Independent Research Program of Tsinghua University (20111080998), and the China Postdoctoral Science Foundation (No. 2014M550068).

References

[1] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, April 2014.

[2] P. Agrawal and S. Rao, "Energy-aware scheduling of distributed systems," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1163–1175, Oct 2014.

[3] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types." in *NSDI*, vol. 11, pp. 24–24, 2011.

[4] D. A. Menascé, V. A. Almeida, and L. W. Dowdy, *Capacity Planning for Web Services: metrics, models, and methods*, Prentice Hall PTR Upper Saddle River, 2002.

[5] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 3, no. 1, pp. 1–39, 2008.

[6] P. Lama and X. Zhou, "Ninepin: Non-invasive and energy efficient performance isolation in virtualized servers," in *the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–12, 2012.

[7] H. Goudarzi and M. Pedram, "Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems," in *IEEE International Conference on Cloud Computing (CLOUD)*, pp. 324–331, 2011.

[8] Z. Zhu, J. Bi, H. Yuan, and Y. Chen, "Sla based dynamic virtualized resources provisioning for shared cloud data centers," in *IEEE International Conference on Cloud Computing (CLOUD)*, pp. 630–637, July 2011.

[9] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters," in *Proceedings of the 6th international conference on Autonomic computing*, pp. 117–126, 2009.

[10] H. Khazaei, J. Misic, and V. B. Misic, "A fine-grained performance model of cloud computing centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 11, pp. 2138–2147, 2013.

[11] P. Padala, K.-Y. Hou, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, and A. Merchant, "Automated control of multiple virtualized resources," in *Proceedings of the 4th ACM European conference on Computer systems*, pp. 13–26, 2009.

[12] R. Ghosh, K. S. Trivedi, V. K. Naik, and D. S. Kim, "End-to-end performability analysis for infrastructure-as-a-service cloud: An interacting stochastic models approach," in *IEEE 16th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 125–132, 2010.

[13] K.-J. Bathe and E. L. Wilson, "Numerical methods in finite element analysis," 1976.

[14] D. Gross, *Fundamentals of queueing theory*, John Wiley & Sons, 2008.

[15] J. McKenna, "A generalization of little's law to moments of queue lengths and waiting times in closed, product-form queueing networks," *Journal of Applied Probability*, pp. 121–133, 1989.

[16] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 123–134, 2009.

[17] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, pp. 13–23, 2007.

[18] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, Springer, pp. 760–766, 2010.

[19] L. Ingber, A. Petraglia, M. R. Petraglia, M. A. S. Machado *et al.*, "Adaptive simulated annealing," in *Stochastic global optimization and its applications with fuzzy adaptive simulated annealing*, Springer, pp. 33–62, 2012.

[20] C. Amza, A. Chanda, A. L. Cox, S. Elnikety, R. Gil, K. Rajamani, W. Zwaenepoel, E. Cecchet, and J. Marguerite, "Specification and implementation of dynamic web site benchmarks," in *IEEE International Workshop on Workload Characterization*, pp. 3–13, 2002.

[21] H. W. Cain, R. Rajwar, M. Marden, and M. H. Lipasti, "An architectural evaluation of java tpc-w," in *The Seventh International Symposium on High-Performance Computer Architecture*, pp. 229–240, 2001.

[22] M. Arlitt and T. Jin, "A workload characterization study of the 1998 world cup web site," *IEEE Network*, vol. 14, no. 3, pp. 30–37, 2000.

[23] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, 2011.

[24] X. Deng, D. Wu, J. Shen, and J. He, "Eco-Aware Online Power Management and Load Scheduling for Green Cloud Datacenters," *IEEE Systems Journal*, pp. 1–10, 2014. doi=10.1109/JSYST.2014.2344028

[25] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164–177, 2003.

[26] S. Agrawal, Y. Dashora, M. K. Tiwari, and Y.-J. Son, "Interactive particle swarm: a pareto-adaptive metaheuristic to multiobjective optimization," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 2, pp. 258–277, 2008.