Application-Aware Dynamic Fine-Grained Resource Provisioning in a Virtualized Cloud Data Center

Jing Bi, *Member, IEEE*, Haitao Yuan, *Student Member, IEEE*, Wei Tan, *Senior Member, IEEE*, MengChu Zhou, *Fellow, IEEE*, Yushun Fan, Jia Zhang, *Senior Member, IEEE*, and Jianqiang Li

***Abstract*—A key factor of win–win cloud economy is how to trade off between the application performance from customers and the profit of cloud providers. Current researches on cloud resource allocation do not sufficiently address the issues of minimizing energy cost and maximizing revenue for various applications running in virtualized cloud data centers (VCDCs). This paper presents a new approach to optimize the profit of VCDC based on the service-level agreements (SLAs) between service providers and customers. A precise model of the external and internal request arrival rates is proposed for virtual machines at different service classes. An analytic probabilistic model is then developed for non-steady VCDC states. In addition, a smart controller is developed for fine-grained resource provisioning and sharing among multiple applications. Furthermore, a novel dynamic hybrid metaheuristic algorithm is developed for the formulated profit maximization problem, based on simulated annealing and particle swarm optimization. The proposed algorithm can guarantee that differentiated service qualities can be provided with higher overall performance and lower energy cost. The advantage of the proposed approach is validated with trace-driven simulations.**

***Note to Practitioners*—Resource allocation plays an important role in constructing scalable and green VCDC. This work presents a novel and fundamental framework to achieve dynamic fine-grained resource allocation. It develops a dynamic fine-grained resource allocation model with non-steady states according to the external and internal workload of different resource-intensive applications in a VCDC. In order to meet the SLA requirements of Gold and Silver services for various applications while maximizing profit, this work proposes a dynamic hybrid optimization algorithm by combing particle swarm optimization and simulated annealing. The experimental results show that the proposed method has a great potential to maximize the VCDC provider's profit. The proposed framework can aid the design and optimization of industrial cloud data centers and practitioners' understanding of SLA aspects of various applications.**

***Index Terms*—Data center, dynamic resource provisioning, heuristic algorithm, optimization.**

Manuscript received August 09, 2015; revised November 01, 2015; accepted November 16, 2015. Date of publication December 28, 2015; date of current version April 05, 2017. This paper was recommended for publication by Associate Editor C.-H. Chen and Editor L. Shi upon evaluation of the reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grants 61033005 and 61174169, and in part by the National Science and Technology Support Program of China under Grant 2012BAF15G01. *(Corresponding author: Haitao Yuan.)*

J. Bi and J. Li are with the School of Software Engineering, Beijing University of Technology, and the Beijing Engineering Research Center for IoT Software and Systems, Beijing 100124, China (e-mail: bijing@bjut.edu.cn; lijianqiang @bjut.edu.cn).

H. Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: cityu.yuan@gmail.com; yuanhaitao@buaa.edu.cn).

W. Tan is with the IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: wtan@us.ibm.com).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA (e-mail: zhou@njit.edu).

Y. Fan is with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: fanyus@tsinghua.edu.cn).

J. Zhang is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: jia.zhang@sv.cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2015.2503325

I. Introduction

With the wide deployment of cloud computing services, virtualized cloud data centers (VCDCs) have become increasingly more important. Various applications concurrently running in VCDCs, are intensive on different resources, such as CPU- and I/O-intensive ones (i.e., computing-intensive and data-intensive applications), thus require various infrastructure resources [1]. Traditional resource allocation for a single type of resource intensive application is inefficient, since it can lead to much resource waste. For example, compute (or CPU)-intensive applications may occupy CPU resources for a long time, while wasting I/O resources in a physical machine (PM) or virtual machine (VM). Note that in this work, I/O refers to local disk I/O excluding network-based storage such as the elastic block store and simple storage service. Moreover, due to the increasing energy cost associated with data centers [2], [3], it will be too costly 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 multiresource sharing among various types of applications including computing-intensive and data-intensive jobs. A number of dynamic resource provisioning methods, such as round-robin [4], control-theoretic [5], and machine-learning [6], have been proposed for effective allocation of such resources as CPU, memory, storage, and network bandwidth to various applications.

Unfortunately, most existing methods fail to realize the objectives to minimize service provider's energy cost and maximize revenue in complex cloud environments. They mainly focus on a single type of resources even in multiresource environments where customers have heterogeneous resource requirements. For example, a Hadoop scheduler [7] always allocates the fixed-size partition of PM resources to different requests in these clusters. It ignores the fact that the requests may have varying demands for CPU, memory, and I/O resources. In recent years, the virtualization technology consolidates multiple online application services into fewer physical resources. Based on

virtualization, some 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 offline capacity planning process.

This study provides a way to allocate various heterogeneous resources to requests from different applications in a virtualized cloud data center (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. Here, there is a tradeoff between a VCDC provider's energy cost and revenue. Our method focuses on the *profit maximization* problem and provides dynamic fine-grained resource provisioning while meeting the SLA demands of different resource-intensive applications in a VCDC.

To summarize, our contributions in this work are threefold.

1) We accurately compute request arrival rates based on the external and internal workload for resource-intensive applications, and establish an analytic probabilistic system model to deal with *non-steady states* in a VCDC.
2) We formulate the profit maximization problem as a mixed integer nonlinear programming (MINLP) and propose a *dynamic hybrid provisioning* algorithm to solve it.
3) We propose a novel *Smart Controller* (SC) to support a dynamic fine-grained resource provisioning based on the SLA demands of heterogenous applications, and evaluate its effectiveness via trace-driven simulation.

The rest of this 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 multiple resources, and proposes a solution algorithm. Section VI presents the performance evaluation results. Section VII concludes this paper.

II. Related Work

A. Dynamic Resource Allocation

As a fundamental problem in VCDCs, resource allocation aims to provision limited resources while guaranteeing the arrival performances of customer requests. Recently, a number of methods on resource allocation in VCDCs have been proposed [8]–[12]. For example, Xiong *et al.* address the issue of how to intelligently manage the resources in a shared cloud database system and present SmartSLA, a cost-aware resource management system. SmartSLA consists of the system modeling module and the resource allocation decision module, and adjusts the resource allocations to achieve the optimum profit [10]. However, they only consider the action cost related to database systems. Xia *et al.* propose a stochastic model and quality evaluation approach for Infrastructure-as-a-Service cloud by considering expected request completion time, rejection probability, and system overhead rate as key quality metrics [11]. According to [12], compared with a dynamic resource provisioning method, a static one cannot dynamically serve varying workload demands while meeting contracted SLA guarantees. However, they use static linear models that are obtained by a system identification method where the parameters are identified offline. In general, the above researches fail to provide an appropriate provisioning approach for end-to-end SLAs. Moreover, they cannot be directly applied to resource allocation for various kinds of applications. In contrast, this paper proposes a VM profit model for the fine-grained sharing of physical infrastructure according to the request arrival rates of computing-intensive and data-intensive applications.

B. Virtualized Resource Allocation

Virtualization technology is an efficient resource sharing approach to support various applications in VCDCs [13]–[20]. 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, Menascé *et al.* consider the issue of autonomically allocating CPU resources to various VMs as workload varies [15]. In order to do so dynamically, Kalyvianaki *et al.* adopt Kalman filters to track and control CPU utilization in virtualized environments [16]. Khazaei *et al.* propose models to consider the important features of cloud centers such as batch arrival of user requests, resource virtualization, and realistic servicing steps, to obtain important performance metrics including task blocking probability and total waiting time incurred on user requests [17]. In contrast to these researches, our work can support a fine-grained and heterogeneous resource allocation for a virtualized cloud computing environment.

Garg *et al.* tackle the resource allocation problem within a datacenter that runs different types of workloads, particularly non-interactive and transactional applications. They propose admission control and scheduling mechanism which not only maximizes the resource utilization and profit, but also ensures the SLA requirements of users [18]. Padala *et al.* present a resource control system that achieves application performances by automatically adapting to dynamic workload changes [19]. They provide an MIMO resource controller to manage multiple resources. Zhu *et al.* propose a resource provisioning method with budget constraints for a class of adaptive applications in cloud environments [20]. Different from theirs, our method cannot only provide differentiated service qualities but also reduce energy cost. Moreover, our method can accurately compute request arrival rates according to different resource-intensive applications in a VCDC.

C. SLA-Based Non-Steady State Allocation

Recently, a few studies focus on SLA resource allocation issues for data centers. However, they cannot be readily adapted to cloud computing environments because they usually assume equilibrium states and adopt mean value analysis [21]–[23]. For example, Urgaonkar *et al.* present an analytical model of dynamic resource provisioning for multitier clusters [21]. 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 [22]. 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 [23]. They consider CPU, memory and network resource requirement. However, their single and simple M/M/1 queueing system cannot reflect real cases well.

Different from the prior work, 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 archived VCDC performance.

III. Motivation and System Architecture

While previous work on resource allocation focused on single type of resources or some particular certain applications, the advent of cloud computing environment and multicore processors enables one to meet heterogeneous application demands. In a traditional data center, each PM can only serve one application at a time. By contrast, in a VCDC, when a service request is processed, a prebuilt image is used to create one or more VM instances. When the VM instances are deployed, they are provisioned with specific CPU, memory, and disk I/O capacity. VMs are deployed on PMs, each of which may be shared by multiple VMs.

Existing schedulers for data centers ignore the SLA demands of heterogeneous applications. This leads to inefficient and sometimes infeasible resource allocation to meet different application demands. For example, the requests of computing-intensive applications typically occupy CPU resources for a long time. This may waste a large amount of I/O resources allocated to them. Therefore, it is meaningful to satisfy the SLAs of resource-intensive applications by allocating their corresponding types of virtualized resources. In this way, we can improve the overall resource utilization in a VCDC, and reduce the unnecessarily occupied resources. Meanwhile, we can ultimately maximize a VCDC provider's profit.

One particular motivation of this work is due to the clear need to pack together applications with complementary multiresource allocation requirements, such as placing a compute (or CPU)-intensive VM and a data (or I/O)-intensive VM on the same PM. We propose a *Smart Controller* (SC) architecture, as shown in Fig. 1. In this work, to improve a VCDC performance and maximize its profit, we use the SC to support a dynamic fine-grained resource provisioning according to the SLA demands of heterogeneous applications. We accurately compute request arrival rates based on the external and internal workload for resource-intensive applications, and establish an analytic probabilistic system model to deal with non-steady states in SC. SC can also minimize the consumption of computing and storage resources by specifying the number of PMs and VMs, as well as CPU and local disk I/O shares per VM at each control interval.

Moreover, we assume that PMs and VMs are categorized into three states: hot (i.e., powered on with running), warm (powered on, but not ready), and cold (powered off) [24]. Powered on PMs and VMs are placed in hot and warm clusters, while powered off ones are placed in cold clusters to reduce energy consumption during the periods of small workload. Due to different performance levels of various applications, in our work, Gold services pose higher performance requirement than silver ones. Besides, the operating systems are encapsulated into each separated VM. Thus, multiple VMs can be used for parallel processing of various applications.

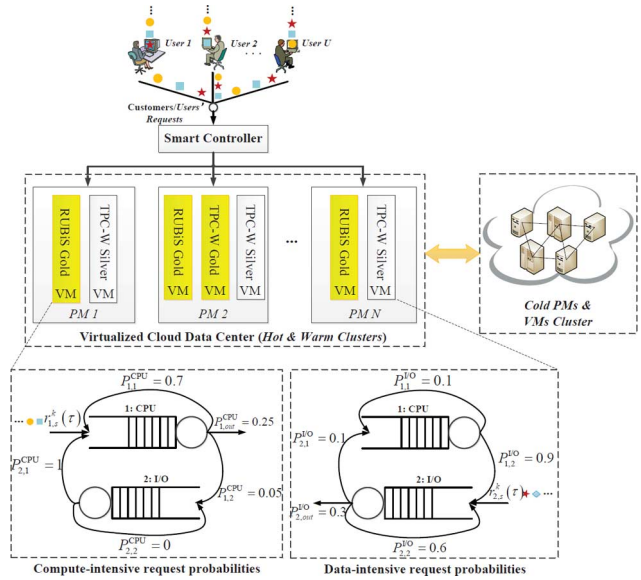


Fig. 1. System architecture.

Based on the proposed architecture, Section IV constructs a system model. Section V formulates a dynamic multiple resource provisioning problem with the purposes of maximizing the profit, and gives an optimization algorithm to solve it.

IV. System Model

The arrival rates of service requests for a wide variety of applications can vary from time to time at a VCDC. To improve a VCDC performance and maximize its profit, we first develop a system model. Our control objective is to maximize the profit generated by Gold and Silver services for resource-intensive applications under a time-varying workload by dynamically tuning the following parameters at the beginning of each control interval. Parameters are summarized in Table I.

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

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

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

Its state of the sth service class is denoted as

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

TABLE I
SYMBOLS USED IN FORMULATION

Model parameters	
I	Number of resource types, i=1, ..., I
K	Number of intensive applications, k=1, ..., K, where ⟨1,2⟩=⟨CPU, I/O⟩=⟨Compute, Data⟩
S	Number of service classes, s=1, ..., S, where ⟨1,2⟩=⟨Gold, Silver⟩
$q_{i,s}^k(\tau)$	Expected number of queued requests of the k-th intensive application for the s-th service class into the i-th resource type
$\Lambda_{i,s}(\tau)$	Total average internal and external arrival rate for the s-th service class into the i-th resource type
$r_{i,s}^k(\tau)$	External arrival rate of the k-th intensive application for the s-th service class into the i-th resource type
$\lambda_{i,s}^k(\tau)$	Average arrival rate of the k-th intensive application for the s-th service class into the i-th resource type
$\mu_{i,s}^k(\tau)$	Service rate of the k-th intensive application for the s-th service class into the i-th resource type
$\gamma_s^{k,\mathbb{C}}(\tau)$	Number of finished requests in SLA of the k-th intensive application for the s-th service class
$\gamma_s^{k,\mathbb{N}}(\tau)$	Number of rejected requests in SLA of the k-th intensive application for the s-th service class
$E(T_s^k(\tau))$	Expected average response time of the k-th intensive application for the s-th service class
$E(T_{i,s}(\tau))$	Expected average response time of i-th resource type for the s-th service class
T	Sampling period of SC
N_{max}	Maximum number of PMs (n=1, ..., N_{max})
V_{max}	Maximum number of VMs (v=1, ..., V_{max})
Control variables	
$N(\tau)$	Number of powered on PMs
$V(\tau)$	Number of powered on VMs
$\varphi_s^k(\tau)$	Number of hot PMs ($\varphi \leq N$) of k-th intensive application for s-th service class
$c_s^k(\tau)$	Number of hot VMs ($c \leq V$) of the k-th intensive application for the s-th service class
$\phi_{s,v}^k(\tau)$	CPU allocation of the v-th VM to the k-th intensive application for the s-th service class
$\theta_{s,v}^k(\tau)$	I/O allocation of the v-th VM to the k-th intensive application for the s-th service class
$\omega_{s,v}^k(\tau)$	Fraction of the k-th intensive application for the s-th service class workload to the v-th VM
SLA parameters	
$\overline{T}_s^k$	The target response time of the k-th intensive application for the s-th service class in SLA Deadline
$\mathbb{C}_s^k$	Unit request revenue of the k-th intensive application for the s-th service class in SLA
$\mathbb{N}_s^k$	Unit request refund of the k-th intensive application for the s-th service class in SLA

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 ith 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 kth intensive application for the sth service class, respectively.

Its control variable related to the sth service class is denoted as

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

where $N(\tau)$ and $V(\tau)$ are the number of powered 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 vth VM in the kth intensive application for the sth service class, respectively. $\omega_{s,v}^k(\tau)$ is workload fraction directed to the vth 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 workload arrival rate of the kth intensive application for the sth service class into the ith 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, and we adopt the Gauss-Seidel iterative method [25] to approximate $\lambda_{i,s}^k$ and $\lambda_{j,s}^k \cdot P_{(j,s),(i,s)}^k$ denotes the probability that when a kth intensive application finishes at resource type j, it next moves to resource type i for the sth service class.

Then, the total average arrival rate from both the internal and external of all intensive applications for the sth service class into the ith 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 kth intensive applications for the sth service class into resource type i (j), respectively.

Thus, the total average arrival rate from both the internal and external of all intensive applications for the sth service class into the ith 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 powered on VMs, and CPU and disk I/O allocations given to each VM at time τ. In this paper, we assume that hot PMs and VMs equal to powered on PMs and VMs, respectively. Each VM is assigned a share of the PM's CPU, memory, and local 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, time interval is β, e.g., 30 s, and total t interval times in sampling period, that is, $T = t \cdot \beta$, 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 in sampling period. 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 $\mu_{i,s}^k$. It can be obtained by

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

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 ith type resource is enough to process all requests of the kth intensive application for the sth 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 subcases: 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 [14]; 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 [26] 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 [27], 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 $E(T_s^k)$ denote the total expected average response time in a VCDC of the kth intensive application for the sth service class. We use subscript $\tilde{k}$ to show resource type k, and superscript k the kth 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 sth service class. $P_{\tilde{k},out}^k$ means the probability that the kth 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 kth 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 and switching on VMs is ignorable. In addition, CPU and local disk I/O resource overheads in VM migration may further degrade application performance on an already congested node. Therefore, VM migration is mainly effective for sustained rather than transient overload. We also assume that the time for VM migration is ignorable.

B. Energy Consumption

In order to reduce CDC'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 CDC. However, a CDC typically runs multiple VMs on each PM. It is thus highly desired to pack together requests with complementary resource requirements. This work focuses on critical machine-level energy consumption, and therefore does not directly account for CDC-level energy conversion loss and the energy used for cooling infrastructure. 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 [28]. 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 power 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 kth intensive application for the sth service class at time τ. F, A, and ϵ are the fixed power, variable power, and empirically derived correction constant, respectively [29]. 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 kth intensive application for the sth service class. U is the power usage effectiveness of a VCDC. From (16), the energy consumption at a VCDC increases as we power 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 $E(T_s^k) \le \overline{T}_s^k$, "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

$$Revenue = \begin{cases} \sum_{k=1}^{K} \sum_{s=1}^{S} \mathbb{C}_s^k \cdot \gamma_s^{k,\mathbb{C}} & \text{if } E(T_s^k) \le \overline{T}_s^k \\ \sum_{k=1}^{K} \sum_{s=1}^{S} \mathbb{C}_s^k \cdot \gamma_s^{k,\mathbb{C}} - \mathbb{N}_s^k \cdot \gamma_s^{k,\mathbb{N}} & \text{otherwise} \end{cases} \tag{20}$$

where $\mathbb{C}_s^k$ and $\mathbb{N}_s^k$ are the unit request revenue and refund of the kth intensive application for the sth service class, respectively. According to the predicted result of the system metrics, we can conclude that if $\lambda_{i,s}^k < \mu_{i,s}^k$, $\gamma_s^{k,\mathbb{C}} = \lambda_{i,s}^k \cdot P(E(T_s^k) \le \overline{T}_s^k) \cdot T$, 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 within sampling period T for the kth intensive application of the sth service class, respectively. Otherwise, if $\lambda_{i,s}^k \ge \mu_{i,s}^k$, 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 $\Lambda_{i,s}^{k*}$. Therefore, the number of finished and rejected requests within sampling period T for the kth intensive application of the sth service class is $\gamma_s^{k,\mathbb{C}} = \Lambda_{i,s}^{k*} \cdot P(E(T_s^k) \le \overline{T}_s^k) \cdot T$ and $\gamma_s^{k,\mathbb{N}} = \left[\lambda_{i,s}^k - \Lambda_{i,s}^{k*} \cdot P(E(T_s^k) \le \overline{T}_s^k)\right] \cdot T$, respectively. $\mathbb{C}_s^k \cdot \gamma_s^{k,\mathbb{C}}$ denotes the total revenue received by a VCDC for the requests of the kth intensive application for the sth service class that are handled before an SLA-deadline. $\mathbb{N}_s^k \cdot \gamma_s^{k,\mathbb{N}}$ denotes the total refund paid to customers for the requests of the kth intensive application for the sth 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 $P(E(T_s^k) \le \overline{T}_s^k) \cdot \lambda_{i,s}^k$ or $P(E(T_s^k) \le \overline{T}_s^k) \cdot \Lambda_{i,s}^{k*}$ service requests per second. A VCDC's average CPU utilization at time τ can be obtained as:

1) underloaded:

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

2) overloaded:

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

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

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

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 VCDC's energy expenditure and its revenue for various resource-intensive applications. Our proposed SC's goal is to find optimal CPU and I/O resource allocations of N PMs for the set of V VMs while maximizing the profit of a VCDC.

Therefore, based on the given profit function in (19), the final profit maximization problem (PMP) can be summarized as follows:

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

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

$$\max_{\eta,\sigma} \sum_{y=\tau+1}^{\tau+h} Profit(\eta(y), \sigma(y)) = \max_{\eta,\sigma} \{f_1 - f_2\}$$

s.t.

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

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

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

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

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

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

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

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

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., $h = 5, T = 30, h \cdot T = 150$ 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 τ. 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 $\omega_{s,v}^k(y)$ is workload fraction directed to the vth VM. The control variable $d_{s,v,n}^k(y) \in \{0,1\}$ indicates whether the vth VM of the kth intensive application for the sth service class is allocated to PM $n \in \{1,\ldots,N_{\max}\}$. 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 $E(T_s^k(y))$ cannot exceed the target response time $\overline{T}_s^k$ of the kth intensive application for the sth service class specified in SLA. Constraint (32) shows that the situation of steady and non-steady states.

B. Solution Algorithm

We first apply the method of a penalty function to convert constrained problem PMP into an unconstrained one (UPMP). Each equality or inequality constraint introduces a penalty to the objective function in PMP. For an optimization problem with $\mathbb{A}$ equality constraints and $\mathbb{B}$ inequality constraints, each equality constraint $h_\alpha(\vec{x}) = 0 (1 \le \alpha \le \mathbb{A})$ brings the penalty of $|h_\alpha(\vec{x})|^\nu$ while each inequality constraint $g_\beta(\vec{x}) \ge 0$ $(1 \le \beta \le \mathbb{B})$ brings the penalty of $[\max(0, -g_\beta(\vec{x}))]^\partial$. Let Γ and $\vec{x}$ denote the total penalty brought by all constraints and the vector of control variables, respectively. Γ can be calculated as

$$\Gamma = \sum_{\alpha=1}^{\mathbb{A}} |h_\alpha(\vec{x})|^\nu + \sum_{\beta=1}^{\mathbb{B}} [\max(0, -g_\beta(\vec{x}))]^\partial. \tag{33}$$

Therefore, the unconstrained problem UPMP is shown as

$$\min_{\eta,\sigma}\left[\Omega\cdot\Gamma - \left[\sum_{y=\tau+1}^{\tau+h} Profit(\eta(y),\sigma(y))\right]\right] \tag{34}$$

where the parameter Ω is a relatively big positive constant.

Based on the optimization problem formulated in the above section, this subsection presents a solution algorithm for the problem which is used by SC. Note that in the problem, objective functions f_1 and f_2 are nonlinear. Besides, N, V, φ_s^k and c_s^k are integer variables, while $\phi_{s,v}^k, \theta_{s,v}^k$ and $\omega_{s,v}^k$ are continuous ones. The optimization problem contains both discrete and continuous variables, and involves nonlinear objective function. Therefore, the formulated problem is a mixed integer nonlinear programming (MINLP) [30], which is NP-complete.

Several present algorithms have been proposed to tackle MINLP, e.g., equality relaxation [31], and branch and bound [32]. However, they usually rely on the problem specific structure based on which the original problem is converted into another particular problem that can be solved in an easier way. They converge to the global optimum but at the expense of sometimes unacceptably long execution time. Existing metaheuristic methods can prevent drawbacks of the above algorithms, and are robust to solve many optimization problems with different types of mathematical structures. Though they do not guarantee the globally optimal solution, they can be easily implemented, and therefore they have been commonly applied in solving complex MINLP problems. However, different metaheuristic algorithms exhibit their respective strengths and weaknesses. For example, particle swarm optimization (PSO) can obtain solutions in a quicker way, but easily trap into local optima when it is adopted to tackle MINLP according to [9]. Simulated annealing (SA) can converge to the global optima by accepting worse solutions according to the Metropolis criterion [33]. However, its convergence speed is relatively slow especiallyfor complex MINLP with a large search space.

To ensure a timely solution with acceptable quality, which is critical in a VCDC, we propose a hybrid metaheuristic algorithm, HSPA, to solve the proposed problem. This work combines the strengths of PSO and SA for solving MINLP while keeping some speed advantage. In the proposed algorithm, each particle dynamically updates its current position based on other particles' and its own positions. Different from PSO, the old and new positions of each particle are compared in terms of an objective function. In each iteration, better solutions are directly accepted into the next generation while worse ones are accepted to escape from local optima and finally obtain global optima. Therefore, the proposed algorithm can increase the possibility of finding the globally optimal solution that can maximize the profit of a VCDC.

The profit maximization problem is solved by HSPA shown in Algorithm 1. We first introduce the notations in HSPA. w denotes the inertia weight used to limit the variation of each particle's velocity. $w_{\max}$ and $w_{\min}$ denote the upper and lower bound of inertia weight, respectively. o_1 denotes the local acceleration coefficient that represents individual search ability of each particle. o_2 denotes the social acceleration coefficient that shows the global search ability of each swarm. To prevent unexpected roaming positions of particles, the range of each particle's velocity is limited to $[-v_{\max}, v_{\max}]$. gB denotes the best position of all particles in current swarm. In addition, pB denotes the best position of every particle in current swarm. Besides, cr and tmp^0 denote the cooling rate of temperature and

the initial temperature, respectively. $\eth$ and ξ denote the total number of iterations, and the size of the swarm, respectively.

Algorithm 1. HSPA

1: **if** $\tau = 0$ **then**
2: Randomly initialize position and velocity of each particle in PS_0^τ of size ξ
3: **else**
4: $PS_0^\tau \leftarrow PS_{\eth}^{\tau-1}$
5: **end if**
6: Update the fitness values of all particles in PS_0^τ
7: Update pB and gB
8: Initialize $w_{\min}$, $w_{\max}$, o_1, o_2, $v_{\max}$, cr and tmp
9: $w \leftarrow w_{\max}$
10: $tmp \leftarrow tmp^0$
11: $z \leftarrow 0$
12: **while** $z \le \eth$ and $\varpi \le 95\%$ **do**
13: Update positions and velocities of all particles according to the acceptance criterion of Metropmolis
14: Calculate fitness values of all particles in PS_z^τ
15: Update pB and gB
16: $PS_{z+1}^\tau \leftarrow PS_z^\tau$
17: $tmp \leftarrow tmp \cdot cr$
18: $w \leftarrow w_{\max} - (w_{\max} - w_{\min}) \cdot z/\eth$
19: $z \leftarrow z + 1$
20: **end while**
21: Output gB

In Algorithm 1, Line 2 randomly initializes the first swarm if $\tau = 0$. Otherwise, Line 4 shows that positions and velocities of all particles in PS_0^τ are initiated with $PS_{\eth}^{\tau-1}$. Line 6 updates the fitness values of all particles in PS_0^τ based on (34). Then, Line 7 updates pB and gB. Lines 9–10 initiate the inertia weight and the temperature with $w_{\max}$, and tmp^0, respectively. The **while** loop is shown in Lines 12–20. Line 13 updates positions and velocities of all particles in PS_z^τ according to the acceptance criterion of Metropolis. Lines 14–15 evaluate the fitness values of all particles in PS_z^τ, and update pB and gB, respectively. Then, the next swarm in time τ, PS_{z+1}^τ, is generated in Line 16. Lines 17–18 update the temperature, tmp, and inertia weight, w, respectively. Let ϖ denote the percentage of particles with the same fitness value in PS_z^τ. The **while** loop terminates when the number of executed iterations is more than $\eth$ (i.e., $z > \eth$), or the percentage of particles with the same fitness value in PS_z^τ is more than 95% (i.e., $\varpi > 95\%$). Line 21 outputs gB that can be transformed into control variables of the profit maximization problem in time τ.

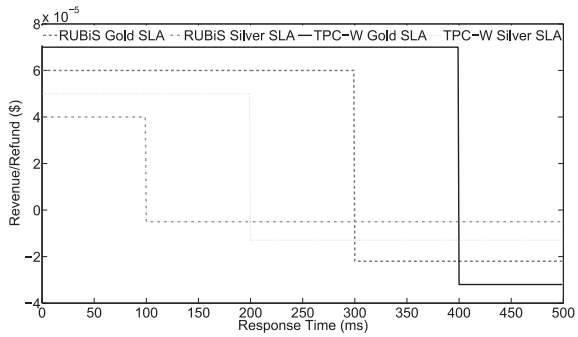


Fig. 2. Service-level agreements (SLAs) in a VCDC.

VI. Performance Evaluation

A. Simulation Setup

Similar to the work [34], this section adopts the trace-driven simulation to evaluate our proposed application-aware dynamic fine-grained resource provisioning in a VCDC. Here, according to the trace analysis of actual network business website [15], the workload arrival rates conform to Poisson distribution. Therefore, we assume that requests arrive in a Poisson process. This means that the interarrival time obeys a negative exponential distribution, which has the Markov characteristics. Every interaction of a customer with the system is a separate request which is independent of others. We adopt two applications about different web-based service classes in our experiments: one is RUBiS [35] that is an online auction site benchmark; the other one is TPC-W [10] that is a transactional web e-Commerce benchmark. The resource requests of RUBiS are different from those of TPC-W in the variations of workloads. For RUBiS, we use a workload mix called the browsing mix that simulates a customer browsing through an auction site. For TPC-W, we use a shopping mix, which simulates a customer browsing through a shopping site. The browsing mix stresses CPU resources, while the shopping mix exerts more demand on I/O resources. Different types of VMs process the corresponding intensive applications across multiple PMs. VMs can share resources (e.g., CPU, memory, and disk I/O) on each PM. Similar to the work [36], we assume that a VCDC offers two service classes, i.e., Gold and Silver services in the proposed SLA model for resource-intensive applications are shown in Fig. 2. Gold and Silver resource-intensive applications contribute revenue based on the nonlinear pricing model that relates the expected average response time of each request to the money that customers are willing to pay. If the response time is below the threshold, a reward is brought to the service provider. Otherwise, a penalty is brought to the service provider.

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 [37] as the service request trend for two service classes of RUBiS, respectively, and 2) the web transaction workload traces from Google's data center [34] for two service classes of TPC-W. Note that RUBiS and TPC-W are CPU-intensive and I/O-intensive applications, respectively, as shown in Fig. 3.

Consider a VCDC with $N_{\max} = 500$ PMs and $V_{\max} = 1500$ VMs, respectively. The exact number of hot PMs and

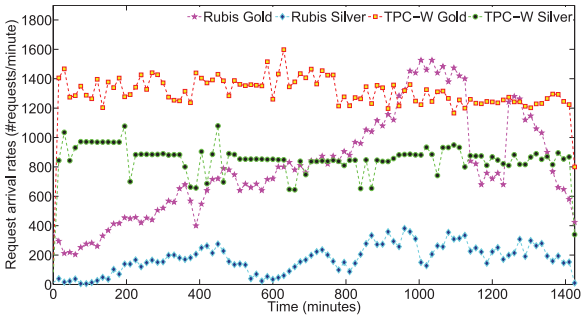


Fig. 3. The workloads of CPU-intensive RUBiS application and I/O-intensive TPC-W application.

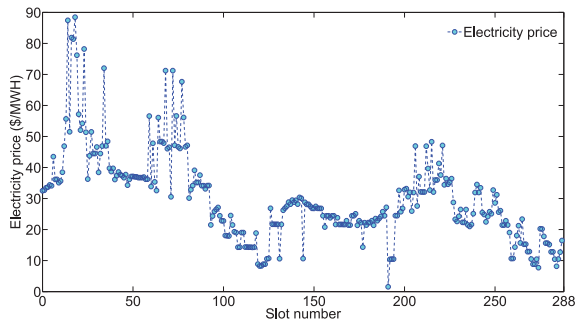


Fig. 4. Electricity price.

VMs are updated periodically at the beginning of each control interval, $h \cdot T$ =150 s, 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 [28]. We assume that the energy model is 65% idle, 1.3 U. The electricity price information is based on the real-time pricing tariffs in Illinois Zone I, on January 1, 2015 [38], as shown in Fig. 4.

In order to evaluate the applicability and effectiveness of the proposed approach in complex cases, we set different values for parameters, as shown in Tables II–V. Two types of resources, i.e., CPU and disk I/O are chosen because they are two most typical configurations in choosing a VM instance for application in a cloud. We assume that the SLA requirements have been specified between two application providers and customers, as is shown in Table II. For example, in the setting of the SLA parameters of two applications, Gold class is the strictest in restriction of the SLA deadline because the corresponding customers pay the most. However, the corresponding refund to customers is also maximum at the same time. VM instance type requested by each application intensive on different resources is selected from the four VM types. According to [39], we set the number of vCPUs to be 10 and 5 for Gold and Silver VMs, respectively. The other parameters of VM instance are set to realistically simulate the situation of resource-intensive applications in Table III. We use the energy model proposed in Section IV-B. We simulate the running cost of the system using a number of different values including the peak server power ($\mathbb{E}_{peak}$), idle server power ($\mathbb{E}_{idle}$) and the U. According to [28], the energy parameters that we use are shown in Table IV: Gold CPU-intensive VM (250 Watts peak, 125 Watts idle, 1.7 U); Silver CPU-intensive VM (240 Watts peak, 120 Watts idle, 1.5 U); Gold I/O-intensive VM (130 Watts peak, 85 Watts idle, 1.3 U); Silver I/O-intensive VM (100 Watts peak, 50 Watts idle, 1.0 U). The probabilities of resource-intensive applications are provided in Table V where RP denotes request probability.

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

B. Analysis and Results

For comparison, we adopted two alternative resource provisioning solutions including *noncapped* [40] and *static* [41] to evaluate the proposed method in a VCDC using trace-driven simulations. We followed the common practice to use the static provisioning as a benchmark to evaluate our methods. Experiments with the same parameter setting are repeated several times. They are based on a discrete-event simulator and resources are allocated periodically. The *noncapped* 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 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. We then establish an analytic probabilistic system model to deal with non-steady states in a VCDC. At the same time, we apply the proposed SC to validate our fine-grained resource provisioning method. Fig. 5 shows the throughputs of RUBiS and TPC-W applications in each control interval of 150 s, respectively. We observed that the throughputs for both applications in our proposed model are close to those in the *noncapped* method, i.e., the actual throughputs. Moreover, the results also demonstrate that our proposed optimization method performs better than the *static* method in terms of throughputs.

Then, we show the results of resource usages for two intensive applications in Fig. 6. We observed 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 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 underutilization or overutilization during the entire execution period. Besides, the *noncapped* 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

TABLE III
PARAMETERS OF CPU AND I/O-INTENSIVE VM INSTANCE TYPES [39]

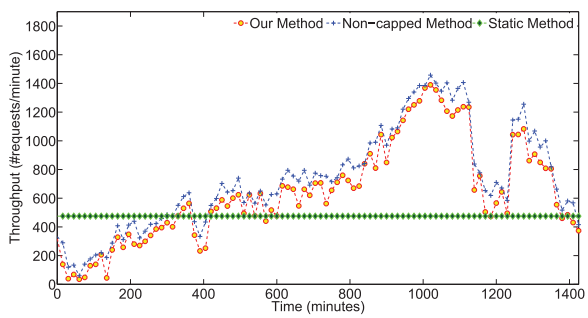
Bounds	Classes	Name	vCPU	CPU Speed (GHz)	Disk (GB)
CPU-intensive VM	1 (Gold)	c3.2Xlarge	10	2.4	2 × 80 SSD
	2 (Silver)	c3.Xlarge	5	2.4	2 × 40 SSD
I/O-intensive VM	1 (Gold)	i3.2Xlarge	10	2.4	2 × 800 SSD
	2 (Silver)	i3.Xlarge	5	2.4	1 × 800 SSD

TABLE IV
ENERGY MODEL PARAMETERS FOR GOLD AND SILVER VMS

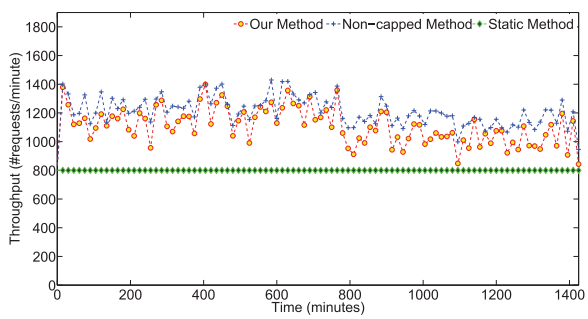
Bounds	Classes	$\mathbb{E}_{peak}$ (Watts)	$\mathbb{E}_{idle}$ (Watts)	U
CPU-intensive VM	1 (Gold)	250	125	1.7
	2 (Silver)	240	120	1.5
I/O-intensive VM	1 (Gold)	130	85	1.3
	2 (Silver)	100	50	1.0

TABLE V
THE REQUEST PROBABILITIES OF RESOURCE-INTENSIVE APPLICATIONS

Bounds	$P^k_{\tilde{k},out}$	$P^k_{\tilde{k},\tilde{k}}$	$P^k_{\tilde{k},i}$	$P^k_{i,\tilde{k}}$
Compute-intensive RPs (%)	0.25	0.7 / 0	0.05	1
Data-intensive RPs (%)	0.3	0.6 / 0.1	0.1	0.9



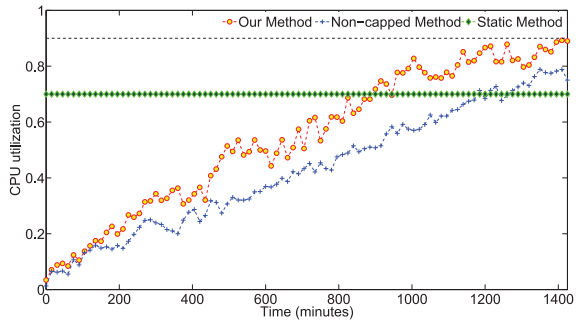
(a)



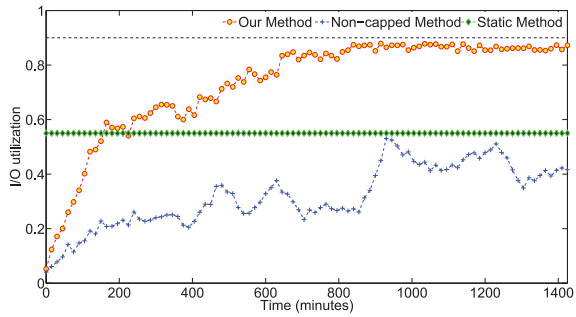
(b)

Fig. 5. Application throughput. (a) RUBiS throughput. (b) TPC-W throughput.

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 set both 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



(a)



(b)

Fig. 6. Resource allocations comparisons on a PM. (a) CPU allocations. (b) Disk I/O allocations.

$h \cdot T = 150$ s. To reduce the frequency of resource reallocations, the control constraints remain unchanged in the control interval.

Our proposed dynamic multiresource hybrid provisioning algorithm is able to maximize the revenue and minimize the energy cost, while meeting all the relevant control constraints. The result illustrated in Fig. 7 shows the accumulated total profit in a VCDC. It can be clearly shown that our method can always perform better than the *noncapped* and *static* methods. The *noncapped* method gives an ideal value for maximizing revenue. Since each resource request is always satisfied, the maximum revenue is achieved. We observed that although the revenue achieved by the *noncapped* 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 *noncapped* 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 *noncapped* method ignores service classes and requested execution time. Hence, high service class requested with long execution time can occupy more resources for processing. Compared with *noncapped* and *static* methods, the number of occupied resources in our model is less. Hence, our proposed method leads to much less energy cost in the

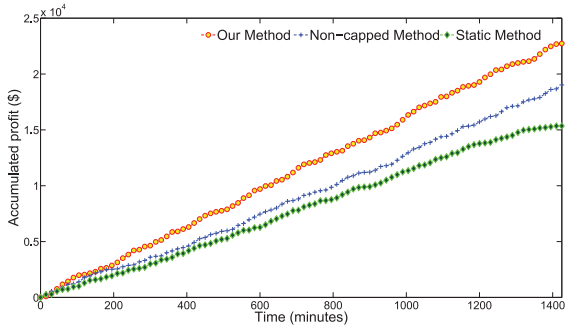


Fig. 7. Accumulated profit.

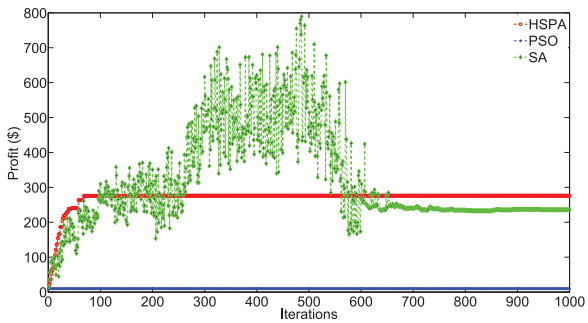


Fig. 8. Evolutionary curves in the 225th minute.

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. We can conclude that our method can realize lower energy cost and achieve higher application profit for various application services in a VCDC.

Fig. 8 further shows the evolutionary curves of the proposed HSPA and baselines including PSO and SA in the 225th minute. The total number of iterations of each algorithm is set to 1000. We can see that among the three algorithms, PSO converges to the final solution after the smallest number of iterations. However, the accuracy of PSO's final solution is the worst because it brings the least profit. The convergence speed of SA is the slowest, and requires 945 iterations to find its final solution. The profit of SA is 24.21 times larger than that of PSO. HSPA only requires 68 iterations to converge to its final solution that brings the profit of 276.06$. Compared with SA, HSPA can bring 40.23$ more profit after much smaller number of iterations.

VII. Conclusion and Future Works

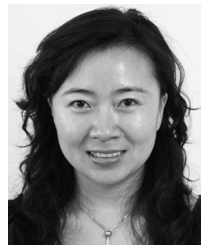
This paper presents a novel analytical model to calculate profit in a virtualized cloud data center. It takes into account several usually ignored factors including the practical service-level agreements that currently exist between cloud providers and their customers, the amount of finished requests, the amount of rejected requests, and price of electricity. We accurately compute request arrival rates based on the external and internal workloads for virtualized cloud data centers, and establish an analytic probabilistic system model to deal with non-steady states in a VCDC for the first time. We then propose a novel smart controller which can realize dynamic fine-grained resource provisioning and manage multiple resource sharing for resource-intensive applications with different service classes. We show that the formulated optimization problem can be formalized as a mixed integer nonlinear program. We propose a particle swarm optimization and simulated annealing combined method to solve it. Finally, the simulation results based on various realistic workload traces have demonstrated the accuracy of the proposed model and effectiveness of the proposed profit maximization method.

We plan to extend our work to investigate how the current approach can be generalized to support resource-intensive applications running in multiple geographically distributed cloud data centers, and apply other resource provisioning methods to profit maximization. In addition, we would like to consider a more general simulation optimization method when a system cannot be accurately modeled by a Markov process. Moreover, network resource is very important to connect servers within a cloud data center [42], [43]. Therefore, we plan to consider the impact of topological structure of networks connecting all nodes in a VCDC.

References

[1] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 564–573, Apr. 2014.
[2] H. Chen, P. Lu, P. Xiong, C. Z. Xu, and Z. Wang, "Energy-aware application performance management in virtualized data centers," *Frontiers of Comput. Sci.*, vol. 6, no. 4, pp. 373–387, Aug. 2012.
[3] P. Agrawal and S. Rao, "Energy-aware scheduling of distributed systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 4, pp. 1163–1175, Oct. 2014.
[4] W. Wang, B. Li, and B. Liang, "Multi-resource round robin: A low complexity packet scheduler with dominant resource fairness," in *Proc. IEEE 21st Int. Conf. Network Protocols*, 2013, pp. 1–10.
[5] Y. Diao, J. Hellerstein, S. Parekh, R. Griffith, G. Kaiser, and D. Phung, "A control theory foundation for self-managing computing systems," *IEEE J. Select. Areas Commun.*, vol. 23, no. 12, pp. 2213–2222, Dec. 2005.
[6] J. Rao, X. Bu, C. Z. Xu, L. Wang, and G. Yin, "VCONF: A reinforcement learning approach to virtual machines auto-configuration," in *Proc. 6th Int. Conf. Auton. Comput.*, 2009, pp. 137–146.
[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in *Proc. 9th USENIX Symp. Netw. Syst. Design Implementation*, 2011, pp. 1–14.
[8] D. Bruneo, "A stochastic model to investigate data center performance and QoS in IaaS cloud computing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 560–569, Mar. 2014.
[9] H. Yuan, J. Bi, W. Tan, and B. Li, "CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers," *IEEE Trans. Autom. Sci. Eng.*, , 2015, doi: 10.1109/TASE.2015.2427234, to be published.
[10] P. Xiong, Y. Chi, S. Zhu, H. J. Moon, C. Pu, and H. HacigümüS, "SmartSLA: Cost-sensitive management of virtualized resources for CPU-bound database services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1441–1451, 2015.
[11] Y. Xia, M. Zhou, X. Luo, Q. Zhu, J. Li, and Y. Huang, "Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 162–170, Jan. 2015.
[12] W. Iqbal, M. Dailey, and D. Carrera, "SLA-driven adaptive resource management for web applications on a heterogeneous compute cloud," in *Proc. IEEE 1st Int. Conf. Cloud Comput.*, 2009, pp. 243–253.
[13] J. Cao, W. Zhang, and W. Tan, "Dynamic control of data streaming and processing in a virtualized environment," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 365–376, Apr. 2012.
[14] J. Bi, H. Yuan, M. Tie, and W. Tan, "SLA-based optimisation of virtualised resource for multi-tier web applications in cloud data centres," *Enterprise Inform. Syst.*, vol. 9, no. 7, pp. 743–767, Sep. 2013.
[15] D. Menasce and M. N. Bennani, "Autonomic virtualized environments," in *Proc. IEEE 2006 Int. Conf. Autonomic Auton. Syst.*, 2006, p. 28.

[16] E. Kalyvianaki, T. Charalambous, and S. Hand, "Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters," in *Proc. 6th Int. Conf. Auton. Comput.*, 2009, pp. 117–126.
[17] H. Khazaei, J. Misic, and V. B. Misic, "A fine-grained performance model of cloud computing centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 11, pp. 2138–2147, Nov. 2013.
[18] S. K. Garg, S. K. Gopalaiyengar, and R. Buyya, "SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter," in *Proc. 11th Int. Conf. Algorithms and Architectures for Parallel Process.*, 2011, pp. 371–384.
[19] 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 *Proc. 4th ACM Eur. Conf. Comput. Syst.*, 2009, pp. 13–26.
[20] Q. Zhu and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments," in *Proc. 19th ACM Int. Symp. High Performance Distrib. Comput.*, 2010, pp. 304–307.
[21] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Trans. Auton. Adaptive Syst.*, vol. 3, no. 1, pp. 1–39, Mar. 2008.
[22] P. Lama and X. Zhou, "NINEPIN: Non-invasive and energy efficient performance isolation in virtualized servers," in *Proc. IEEE/IFIP 42nd Int. Conf. Dependable Syst. Netw.*, 2012, pp. 1–12.
[23] H. Goudarzi and M. Pedram, "Multi-dimensional SLA-based resource allocation for multi-tier cloud computing systems," in *Proc. IEEE Int. Conf. Cloud Comput.*, 2011, pp. 324–331.
[24] 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 *Proc. IEEE 16th Pacific Rim Int. Symp. Dependable Comput.*, 2010, pp. 125–132.
[25] P. W. Wang and C. J. Lin, "Iteration complexity of feasible descent methods for convex optimization," *J. Mach. Learning Res.*, vol. 15, no. 1, pp. 1523–1548, Jan. 2014.
[26] D. Gross, *Fundamentals of Queueing Theory*. New York, NY, USA: Wiley, 2008.
[27] C. J. Kuo, C. F. Chien, and J. D. Chen, "Manufacturing intelligence to exploit the value of production and tool data to reduce cycle time," *IEEE Trans. Autom. Sci. Eng.*, vol. 8, no. 1, pp. 103–111, Jan. 2011.
[28] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 123–134, Oct. 2009.
[29] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *ACM SIGARCH Comput. Architecture News*, vol. 35, no. 2, pp. 13–23, May 2007.
[30] R. C. Baliban, J. A. Elia, R. Misener, and C. A. Floudas, "Global optimization of a MINLP process synthesis model for thermochemical based conversion of hybrid coal, biomass, and natural gas to liquid fuels," *Comput. Chem. Eng.*, vol. 42, pp. 64–86, Jul. 2012.
[31] Y. Lim, K. Jung, and P. Kohli, "Efficient energy minimization for enforcing label statistics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 9, pp. 1893–1899, Sep. 2014.
[32] H. J. Kim, J. H. Lee, and T. E. Lee, "Noncyclic scheduling of cluster tools with a branch and bound algorithm," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 690–700, Apr. 2014.
[33] X. Zhao, "Simulated annealing algorithm with adaptive neighborhood," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1827–1836, Mar. 2011.
[34] Q. Zhang, M. Zhani, R. Boutaba, and J. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, Jan. 2014.
[35] P. Lama and X. Zhou, "Coordinated power and performance guarantee with fuzzy MIMO control in virtualized server clusters," *IEEE Trans. Comput.*, vol. 64, no. 1, pp. 97–111, Jan. 2015.
[36] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: A profit maximization approach," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 1017–1025, Jun. 2013.
[37] M. Arlitt and T. Jin, "A workload characterization study of the 1998 World Cup web site," *IEEE Netw.*, vol. 14, no. 3, pp. 30–37, May 2000.
[38] NYISO, 2015. [Online]. Available: http://www.nyiso.com/public/index.jsp,
[39] Amazon EC2, 2015. [Online]. Available: http://aws.amazon.com/cn/ec2/pricing/
[40] 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 Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, Dec. 2003.
[41] S. Agrawal, Y. Dashora, M. K. Tiwari, and Y. J. Son, "Interactive particle swarm: A Pareto-adaptive metaheuristic to multiobjective optimization," *IEEE Trans. Syst., Man Cybern., Part A: Syst. Humans*, vol. 38, no. 2, pp. 258–277, Mar. 2008.
[42] J. C. Mogul and L. Popa, "What we talk about when we talk about cloud network performance," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 5, pp. 44–48, Oct. 2012.
[43] S. Secci and S. Murugesan, "Cloud networks: Enhancing performance and resiliency," *Comput.*, vol. 47, no. 10, pp. 82–85, Oct. 2014.



Jing Bi (M'13) received the Ph.D. degree from Northeastern University, Shenyang, China, in 2011.

She is currently an Assistant Professor with the School of Software Engineering, Beijing University of Technology, Beijing, China. From 2013 to 2015, she was a Postdoctoral Researcher at the Department of Automation, Tsinghua University, China. From 2009 to 2010, she participated in research on cloud computing at the IBM China Research Laboratory. Her research interests include service computing, cloud computing, large-scale data analytics and resources optimization.

Dr. Bi was the recipient of the 2009 IBM Ph.D. Fellowship Award.

Haitao Yuan (S'15) received the B.S. and M.S. degrees from Northeastern University, Shenyang, China, in 2010 and 2012, respectively. He is currently working toward the Ph.D. degree at the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China.

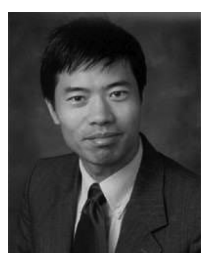
He was a Visiting Scholar with the New Jersey Institute of Technology, Newark, NJ, USA, in 2015. His research interests include cloud computing, resource allocation, and energy efficiency.

Mr. Yuan was the recipient of the 2011 Google Excellence Scholarship.

Wei Tan (M'12–SM'13) received the Ph.D. degree in automation from Tsinghua University, Beijing, China, in 2008.

He is currently a Research Staff Member with the IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. From 2008 to 2010, he was a Researcher with the Computation Institute, University of Chicago, Chicago, IL, USA, and Argonne National Laboratory, Lemont, IL, USA. His research interests include NoSQL, cloud computing, scientific workflows, and Petri nets.

Dr. Tan is member of the Association for Computing Machinery. He is an Associate Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.



MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and is now a Distinguished Professor of Electrical and Computer Engineering. His research interests are in Petri nets, Internet of Things, big data, semiconductor manufacturing, transportation, and energy systems. He has over 600 publications including 12 books, 300+ journal papers (majority in IEEE TRANSACTIONS), and 28 book-chapters. He is the founding Editor of IEEE Press Book Series on Systems Science and Engineering.

Prof. Zhou is a Fellow of the International Federation of Automatic Control and the American Association for the Advancement of Science.

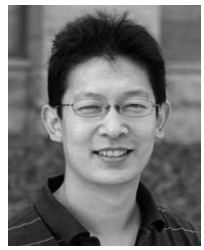
Yushun Fan received the Ph.D. degree in control theory and application from Tsinghua University, Beijing, China, in 1990.

From 1993 to 1995, he was a Visiting Scientist with the Fraunhofer Institute for Production System and Design Technology (FHG/IPK), Germany, supported by Alexander von Humboldt Stiftung. He is currently a Professor with the Department of Automation, Director of the System Integration Institute, and Director of the Networking Manu-Laboratory, Tsinghua University. He has authored ten books in enterprise modeling, workflow technology, intelligent agent, object-oriented complex system analysis, and computer integrated manufacturing. He has published more than 300 research papers in journals and conferences. His research interests include enterprise modeling methods and optimization analysis, business process reengineering, workflow management, system integration and integrated platform, object-oriented technologies and flexible software systems, Petri nets modeling and analysis, and workshop management and control.

Jia Zhang received the M.S. degree in computer science from Nanjing University, Nanjing, China, in 1994, and the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2000.

She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. Her recent research interests center on service oriented computing, scientific workflows, Internet of Things, net-centric collaboration, cloud computing, and big data management. She has coauthored one textbook titled "Services Computing" and has published over 130 refereed journal papers, book chapters, and conference papers.

Prof. Zhang is an Associate Editor of the IEEE TRANSACTIONS ON SERVICES COMPUTING and the *International Journal of Web Services Research*, and Editor-in-Chief of the *International Journal of Services Computing*.



Jianqiang Li received the B.S. degree in mechatronics from the Beijing Institute of Technology, Beijing, China, in 1996, and the M.S. and Ph.D. degrees in control science and engineering from Tsinghua University, Beijing, China, in 2001 and 2004, respectively.

He was a Researcher with the Digital Enterprise Research Institute, National University of Ireland, Galway, from 2004 to 2005. From 2005 to 2013, he was with NEC Labs China as a Researcher, and the Department of Computer Science, Stanford University, as a Visiting Scholar from 2009 to 2010. He joined the Beijing University of Technology, Beijing, China, in 2013 as a Beijing Distinguished Professor. His research interests include Petri nets, enterprise information system, data mining, information retrieval, and big data.