This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

A New Service Mechanism for Profit Optimizations of a Cloud Provider and Its Users

Chubo Liu, Kenli Li, *Senior Member*, *IEEE*, Keqin Li, *Fellow*, *IEEE*, and Rajkumar Buyya, *Fellow*, *IEEE*

Abstract—In this paper, we try to design a service mechanism for profit optimizations of both a cloud provider and its multiple users. We consider the problem from a game theoretic perspective and characterize the relationship between the cloud provider and its multiple users as a Stackelberg game, in which the strategies of all users are subject to that of the cloud provider. The cloud provider tries to select and provision appropriate servers and configure a proper request allocation strategy to reduce energy cost while satisfying its cloud users at the same time. We approximate its servers selection space by adding a controlling parameter and configure an optimal request allocation strategy. For each user, we design a utility function which combines the net profit with time efficiency and try to maximize its value under the strategy of the cloud provider. We formulate the competitions among all users as a generalized Nash equilibrium problem (GNEP). We solve the problem by employing variational inequality (VI) theory and prove that there exists a generalized Nash equilibrium solution set for the formulated GNEP. Finally, we propose an iterative algorithm (IA), which characterizes the whole process of our proposed service mechanism. We conduct some numerical calculations to verify our theoretical analyses. The experimental results show that our IA algorithm can benefit both of a cloud provider and its multiple users by configuring proper strategies.

Index Terms—Cloud computing, Generalized Nash equilibrium, Non-cooperative game theory, Profit optimization, Resource allocation, Variational inequality theory.

1 Introduction

Cloud computing is an increasingly popular paradigm of offering subscription-oriented services to enterprises and consumers [1]. Usually, the provided services refer to Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), which are all made available to the general public in a pay-as-you-go manner [2], [3]. To support various services, more and more cloud centers are equipped with thousands of computing nodes, which results in tremendous energy cost [4]. It is reported that about 50% management budget of Amazon's data center is used for powering and colling the physical servers [5]. There are also researchers who have studied the cost of data centers and concluded that around 40% of the amortized cost of a data center falls into power related categories [6]. Hence, it is important to reduce energy cost for improving the profit of a cloud provider. However, it can often be seen that there are many under-utilized servers in cloud centers, or on the contrary, cloud providers provide less processing capacity and thus dissatisfy their users for poor service quality. Therefore, it is important for a cloud provider to select appropriate servers to provide services, such that it reduces cost as much as possible while satisfying its users at the same time.

For a cloud provider, the income (i.e., the revenue) is the service charge to the aggregated requests from all cloud users [7]. When the per request charge is determined, servers selection and request allocation strategy are two significant factors that should be taken into account. The reason behind lies in that both of them are not just for the profit of a cloud provider, but for the appeals to more cloud users in the market to use cloud service and thus also impact the profit. Specifically, if the provided computing capacity is large enough (i.e., many servers are under-utilized), this will result in tremendous amount of energy waste with huge cost and thus reduces the profit of the cloud provider. On the other hand, if the cloud provider provides less computing capacity or improperly configures the request allocation strategy, this will lead to low service quality (e.g, long task response time) and thus dissatisfies its cloud users or potential cloud users in the market.

A rational user will choose a strategy to use the service that maximizes his/her own net reward, i.e., the utility obtained by choosing the cloud service minus the payment [8]. In addition, the utility of a user is not only determined by the net profit of his/her requests (i.e., how much benefit the user can receive by finishing the configured tasks), but also closely related to the urgency of the tasks (i.e., how quickly they can be finished). The same amount of tasks are able to generate more utility for a cloud user if they can be completed within a shorter period of time in the cloud center [8]. However, considering from energy saving and economic reasons, it is irrational for a cloud provider to provide enough

- *Chubo Liu, Kenli Li, and Keqin Li are with the College of Information Science and Engineering, Hunan University, and National Supercomputing Center in Changsha, Hunan, China, 410082. E-mail: liuchubo@hnu.edu.cn, lkl@hnu.edu.cn, lik@newpaltz.edu.*
- *Keqin Li is also with the Department of Computer Science, State University of New York, New Paltz, New York 12561, USA.*
- *Rajkumar Buyya is with the Department of Computing and Information Systems, The University of Melbourne, Parkville, VIC 3053, Australia. Email: rbuyya@unimelb.edu.au.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

computing resources to complete all requests in a short period of time. Therefore, multiple cloud users have to configure the amount of requests in different time slots. Since the requests from users are submitted randomly, in our paper, we approximately characterize the request arrivals as a Poisson process [9]. Since the payment and time efficiency of each of the cloud users are affected by the decisions of others, it is natural to analyze the behaviors of these users as strategic games [10].

In this paper, we try to design a new service mechanism for profit optimizations of both a cloud provider and its multiple users. We consider the problem from a game theoretic perspective and characterize the relationship between the cloud provider and its users as a Stackelberg game, in which the strategies of all users are subject to that of the cloud provider. In our mechanism, the cloud provider tries to select appropriate servers and configure a proper request allocation strategy to reduce energy cost while satisfying its users at the same time.

The main contributions of this paper are listed as follows.

- We characterize the relationship between the cloud provider and its users as a Stackelberg game, and try to optimize the profits of both a cloud provider and its users at the same time.
- We formulate the competitions among all users as a generalized Nash equilibrium problem (GNEP), and prove that there exists a generalized Nash equilibrium solution set for the formulated GNEP.
- We solve the GNEP by employing varational inequality (VI) theory and propose an iterative algorithm (IA) to characterize the whole process of our proposed service mechanism.

Experimental results show that our IA algorithm can benefit both of the cloud provider and its multiple users by configuring proper strategies.

The rest of the paper is organized as follows. Section 2 presents the related works. Section 3 describes the models of the system and presents the problem to be solved. Section 4 formulates the problem into a Stackelberg game, which consists of a leader and a set of followers. We analyze the strategies for both of the leader and the followers. Many analyses and several subalgorithms are presented in this section. Section 5 is developed to verify our theoretical analysis and show the effectiveness of our proposed algorithm. We conclude the paper with future work in Section 6.

2 Related Work

Some works have been done for profit optimizations of cloud centers in the literature [7], [12], [11], [13]. The methods are presented in Table 1. In [12], Lampe *et al*. proposed a heuristic method to tackle profit maximization for a cloud provider. They focus on auction profit maximization in the context of multiple virtual machines (VMs). In [13], Goudarzi and Pedram developed a heuristic to deal with profit maximization in cloud computing system with service level agreements. They try to reduce cost by powering off appropriate servers, i.e., selecting appropriate servers to provide services. More recently, Cao *et al*. [7] proposed an optimal method for energy saving under continuous dynamic voltage frequency scaling (DVFS) environment. Specifically, they try to configure appropriate speed for each server to save energy. However, as shown in Table 1, all these methods mainly consider from the perspective of the cloud provider.

To our knowledge, hardly any previous works investigate multiple users' profit optimizations, let alone optimizing the profits of a cloud provider and its users at the same time. In this work, we first try to optimize multiple users' profits. Since multiple cloud users compete for using the resources of a cloud provider, and the utility of each user is affected by the decisions (service request strategies) of other users, it is natural to analyze the behaviors of such systems as strategic games [14].

Game theory provides a framework to explain and address the interactive decision situations where the goals and preferences of the participating users are in conflict [15], [16]. It is a formal study of conflicts and cooperation among multiple users [17] and a powerful tool for the design and control of multiagent systems [18]. Due to its advantages, there has been a growing interest in adopting cooperative and non-cooperative game theoretic approaches to various areas such as scheduling [19], communications [20], and evolution of cooperation [21]. A more general framework suitable for investigating and solving various equilibrium models, even when game theory may fail, is the variational inequality (VI) theory which is applicable to a very general class of problems in nonlinear analysis [22]. For more works on game theory, the reader is referred to [23], [24], [25].

As presented in Section 1, energy cost is one of the most important factors that should be taken into account for a cloud provider to increase its profit. Many works have also been done on energy saving in the literature [26], [27], [28], [29]. In [28], Mei *et al*. proposed an energy-aware scheduling algorithm for sporadic tasks. The authors try to reduce energy consumption by using dynamic voltage frequency scaling (DVFS) technique. In [29], based on DVFS technique and the concept of slack sharing among processors, the authors also proposed two novel energy-aware scheduling algorithms. Similar works can also be found in [26], [27].

However, according to [30], even an energy efficient server still consumes about half of its full power when doing no work. Therefore, powering off idle servers when possible is regarded as an effective way to reduce energy cost, especially during off-peak traffic hours for a relative long period of time [31]. In this work, we try to power off some idle servers (i.e., select appropriate servers to provide services) to reduce energy cost for the cloud provider. In addition, we configure server selection strategy for multiple time slots, i.e., for a relative long period of time.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

TABLE 1: Comparison between IA and the state-of-the-art schemes

Schemes	Perspective(s)	Energy saving technique	Characteristic
[7]	Cloud provider	Continuous DVFS	Optimal
[11]	Cloud provider	Powering off servers	Heuristic
[12]	Cloud provider	VM consolidation	Heuristic
[13]	Cloud provider	Powering off servers	Heuristic
IA	Cloud provider and users	Powering off servers	Heuristic

3 System Model and Problem Formulation

In this section, we first present our system models and then formulate the profit optimization problem. We consider the context of a cloud provider with multiple cloud users. The cloud provider is assumed to be equipped with m heterogeneous multicore servers. We denote the set of servers as $\mathcal{M} = \{1, 2, \ldots, m\}$. Each server j ($j \in \mathcal{M}$) consists of c_j cores and similar to [9], it is modeled by an M/M/c queueing system. We denote the set of cloud users as $\mathcal{N} = \{1, 2, \ldots, n\}$. The requests from each of the cloud users are assumed to follow a Poisson process.

We summarize all the notations used in this section in the notation table (see Section 1 of the supplementary material).

3.1 Architecture Model

In this subsection, we model the architecture of our proposed service mechanism, in which the cloud provider can select an appropriate servers subset $\mathcal{S}$ from $\mathcal{M}$ (i.e., $\mathcal{S} \subseteq \mathcal{M}$) to provide services for the H future time slots, and configure a proper strategy $\boldsymbol{p}_{\mathcal{S}} = \left(\boldsymbol{p}_{\mathcal{S}}^1, \ldots, \boldsymbol{p}_{\mathcal{S}}^H\right)$ with $\boldsymbol{p}_{\mathcal{S}}^h = \left(p_j^h\right)_{j \in \mathcal{S}}$ ($h \in \mathcal{H}$) to allocate the aggregated requests to the selected servers, such that the average response time over all cloud users (see Eq. (14)) is minimized, while its multiple users can make an appropriate request decision according to the selected servers and allocation strategy. As shown in Fig. 1, each user i ($i \in \mathcal{N}$) is equipped with a utility function (U_i) and a request configuration strategy ($\boldsymbol{\lambda}_i$), i.e., the request strategy over H future time slots. All requests enter a queue to be processed by the cloud center. Let $\boldsymbol{\lambda}_\Sigma$ be the aggregated request vector, then we have $\boldsymbol{\lambda}_\Sigma = \sum_{i \in \mathcal{N}} \boldsymbol{\lambda}_i$. The cloud provider tries to select an appropriate servers subset $\mathcal{S}$, configure an appropriate allocation strategy $\boldsymbol{p}_{\mathcal{S}}$, and publishes some information (e.g., per request charge r, server subset $\mathcal{S}$, and the corresponding allocation strategy $\boldsymbol{p}_{\mathcal{S}}$, current aggregated requests $\boldsymbol{\lambda}_\Sigma$) on the information exchange model. When multiple users try to configure appropriate request strategies, they first get information from the exchange module, then compute proper request strategies such that their own utilities are maximized and send the newly strategies to the cloud provider.

The computation and communication process can be automatically done by a software. If a user wants to have a look at the aggregated requests $\boldsymbol{\lambda}_\Sigma$ in the process, he/she just needs to press a button of the software to ask for the cloud provider to send the newly updated value of $\boldsymbol{\lambda}_\Sigma$. Take one day as an example, i.e., $H = 24$ (from 20:00 to 20:00 of the next day), with one hour a time slot. The cloud provider sets 20:00 to ensure the users who use its service and compute their corresponding strategies over the next 24 hours. That is to say, each user has two steps to make cloud service reservation. Firstly, before 20:00, the users who want to use the cloud service register their informations. Secondly, the cloud provider collects the informations of its registered users and ensures the agreements at 20:00. If a user registers after 20:00, then he/she tries to make the next negotiation, i.e., waits for the next round.

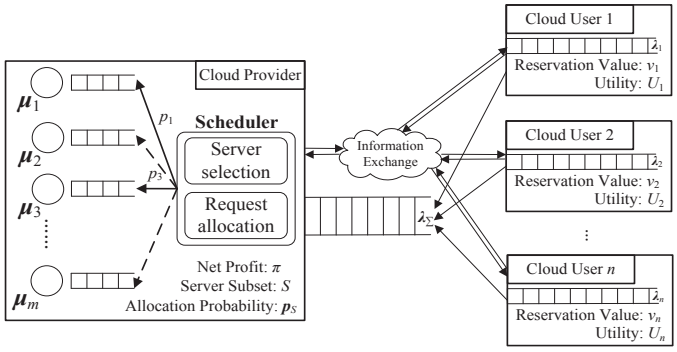


Fig. 1: Architecture model

3.2 Energy Cost Model

We consider energy consumption model in the context of our proposed heterogeneous multicore server system. Energy consumption and circuit delay in complementary metal-oxide semiconductor (CMOS) can be accurately modeled by simple equations, even for complex microprocessor circuits [9]. The energy consumption of a CMOS-based processor is defined as the summation of capacitive, short-circuit, and leakage energy [32]. However, the dominant component in a well-designed circuit is capacitive energy E, which is approximately defined as

$$E = dCV^2 f, \tag{1}$$

where d is the number of switches per clock cycle, C is the total capacitance load, V is the supply voltage, and f is the frequency. The processing capacity of a processor μ is usually linearly proportional to the clock frequency,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

i.e., $\mu \propto f$. With reference to [9], [33], we also obtain $f \propto V^{\phi}$ with $0 < \phi \leq 1$, which implies that $V \propto f^{1/\phi}$. Therefore, we know that the energy consumption is $E \propto f^a$ and $E \propto \mu^a$, where $a = 1 + 2/\phi \geq 3$. In this paper, we assume that

$$E = \xi \mu^a, \tag{2}$$

where ξ is a corresponding factor. Denote χ as the cost of one unit of energy and let E_j be the energy consumption of server j ($j \in \mathcal{M}$) in a unit of time. According to equation (2), we obtain

$$E_j = \chi c_j \xi_j \mu_j^{a_j}, \tag{3}$$

where μ_j is the processing rate of one core of server j, ξ_j and a_j are the corresponding energy consumption factors.

3.3 Request Profile Model

We consider a user request model similar to [34], [35], where the user i's ($i \in \mathcal{N}$) request profile over the H future time slots is formulated as

$$\boldsymbol{\lambda}_i = \left(\lambda_i^1, \ldots, \lambda_i^H\right), \tag{4}$$

where λ_i^h ($h \in \mathcal{H}$) is the arrival rate of requests from user i in the hth time slot and it is subject to the constraint $0 \leq \lambda_i^h \leq \Lambda_i$, where Λ_i denotes user i's maximal requests in a time slot. The requests from each of the users in different time slots are assumed to follow a Poisson process. The individual strategy set of user i can be expressed as

$$\mathcal{Q}_i = \left\{\boldsymbol{\lambda}_i | 0 \leq \lambda_i^h \leq \Lambda_i, \forall h \in \mathcal{H}\right\}, \tag{5}$$

where $\mathcal{H} = \{1, \ldots, H\}$ is the set of all H future time slots.

3.4 Cloud Service Model

The cloud provider is equipped with a request scheduler and m heterogeneous multicore servers. Each server j ($j \in \mathcal{M}$) consists of c_j cores and similar to [9], it is modeled by an M/M/c queuing system. We assume that all of the servers differ in their processing capacities and energy consumptions. The processing capacity of one core of server j ($j \in \mathcal{M}$) is presented by its service rate μ_j. Energy consumption factors ξ_j and a_j are also different among different servers. The cloud provider only selects a servers subset $\mathcal{S}$ ($\mathcal{S} \subseteq \mathcal{M}$) to provide services.

Let p_j^h be the probability that each of the requests is assigned to server j ($j \in \mathcal{S}$) in time slot h ($h \in \mathcal{H}$) and ρ_j^h be the corresponding service utilization. Then we have $\rho_j^h = p_j^h \lambda_\Sigma^h / (c_j \mu_j)$, where λ_Σ^h denotes the aggregated requests from all cloud users in time slot h, i.e., $\lambda_\Sigma^h = \sum_{i=1}^n \lambda_i^h$. Let $\pi_{k,j}^h$ be the probability that there are k service requests (waiting or being proceed) at server j in time slot h. With reference to [9], we have

$$\pi_{k,j}^h = \begin{cases} \pi_{0,j}^h \frac{\left(c_j \rho_j^h\right)^k}{k!}, & k < c_j; \\ \pi_{0,j}^h \frac{c_j^{c_j} \left(\rho_j^h\right)^k}{c_j!}, & k \geq c_j, \end{cases} \tag{6}$$

where

$$\pi_{0,j}^h = \left(\sum_{l=0}^{c_j - 1} \frac{\left(c_j \rho_j^h\right)^l}{l!} + \frac{\left(c_j \rho_j^h\right)^{c_j}}{c_j!} \cdot \frac{1}{1 - \rho_j^h} \right)^{-1}. \tag{7}$$

The probability of queuing (i.e., the probability that a newly submitted request must wait due to all cores of server j are busy) is

$$P_{q,j}^h = \sum_{k=c_j}^{\infty} \pi_{k,j}^h = \frac{\pi_{c_j,j}^h}{1 - \rho_j^h}. \tag{8}$$

The average number of service requests in time slot h (in waiting or in execution) at server j is

$$\bar{N}_j^h = \sum_{k=0}^{\infty} k \pi_{k,j}^h = c_j \rho_j^h + \frac{\rho_j^h}{1 - \rho_j^h} P_{q,j}^h. \tag{9}$$

Applying Little's result, we obtain the average response time at server j as

$$\bar{T}_j^h = \frac{\bar{N}_j^h}{p_j^h \lambda_\Sigma^h} = \frac{1}{p_j^h \lambda_\Sigma^h} \left(c_j \rho_j^h + \frac{\rho_j^h}{1 - \rho_j^h} P_{q,j}^h \right), \tag{10}$$

where $P_{q,j}^h$ represents the probability that the incoming requests at server j need to wait in queue in time slot h. In this paper, we assume that all of the selected servers will likely keep busy, because if not so, some servers could be removed to reduce mechanical wear and energy cost. Therefore, $P_{q,j}^h$ ($\forall j \in \mathcal{S}$) is assumed to be 1, and we have

$$\bar{T}_j^h = \frac{1}{p_j^h \lambda_\Sigma} \left(c_j \rho_j^h + \frac{\rho_j^h}{1 - \rho_j^h} \right) = \frac{1}{\mu_j} + \frac{1}{c_j \mu_j - p_j^h \lambda_\Sigma^h}. \tag{11}$$

With a request rate of λ_i^h ($i \in \mathcal{N}$) in time slot h ($h \in \mathcal{H}$), the average response time of user i on server j ($j \in \mathcal{S}$) is given by

$$\bar{T}_{ij}^h = \frac{p_j^h \lambda_i^h}{\mu_j} + \frac{p_j^h \lambda_i^h}{c_j \mu_j - p_j^h \lambda_\Sigma^h}. \tag{12}$$

We derive the mean response time of user i ($i \in \mathcal{N}$) over all servers as

$$\bar{T}_i^h = \sum_{j \in \mathcal{S}} p_j^h \bar{T}_{ij}^h = \sum_{j \in \mathcal{S}} \left(\frac{\left(p_j^h\right)^2 \lambda_i^h}{\mu_j} + \frac{\left(p_j^h\right)^2 \lambda_i^h}{c_j \mu_j - p_j^h \lambda_\Sigma^h} \right), \tag{13}$$

and the average response time over all users as

$$\begin{aligned} \bar{T}^h &= \sum_{i \in \mathcal{N}} \left(\frac{\lambda_i^h}{\lambda_\Sigma^h} \bar{T}_i^h \right) = \sum_{i \in \mathcal{N}} \left(\frac{\lambda_i^h}{\lambda_\Sigma^h} \sum_{j \in \mathcal{S}} p_j^h \bar{T}_{ij}^h \right) \\ &= \sum_{i \in \mathcal{N}} \frac{\left(\lambda_i^h\right)^2}{\lambda_\Sigma^h} \sum_{j \in \mathcal{S}} \left(\frac{\left(p_j^h\right)^2}{\mu_j} + \frac{\left(p_j^h\right)^2}{c_j \mu_j - p_j^h \lambda_\Sigma^h} \right). \end{aligned} \tag{14}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

3.5 Problem Formulation

Now, let us consider user i's ($i \in \mathcal{N}$) utility in time slot h. A rational cloud user will seek a strategy to maximize its expected net reward by finishing the tasks, i.e., the benefit obtained by choosing the cloud service minus its total payment. We denote user i' net reward in time slot h by R_i^h, where $R_i^h = (b-r)\,\lambda_i^h$ with b and r denoting the benefit factor (the reward obtained by one task request) and the charge factor (the cost by finishing one task request in cloud computing), respectively. On the other hand, since a user will be more satisfied with much faster service, we also take the response time of the user into account. Note that service time utility will be deteriorated with the delay of time slots. Hence, in this paper, we assume that the deteriorating rate of time utility is δ ($\delta > 1$). Denote the $\hat{T}_i^h$ as the time utility of user i in time slot h. Then we have $\hat{T}_i^h = \delta^h \bar{T}_i^h$. More formally, the utility of user i ($i \in \mathcal{N}$) in time slot h is defined as

$$\begin{aligned} U_i^h\left(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h\right) &= w_i R_i^h - \hat{T}_i^h \\ &= w_i\,(b-r)\,\lambda_i^h - \delta^h \bar{T}_i^h, \end{aligned} \tag{15}$$

where $\boldsymbol{\lambda}_{-i}^h = \left(\lambda_1^h, \ldots, \lambda_{i-1}^h, \lambda_i^h, \ldots, \lambda_n^h\right)$ denotes the vector of all users' request profile in time slot h except that of user i, and w_i ($w_i > 0$) is a weight factor, which reflects the importance of net benefit compared with time utility. Note that, when the average response time is low, the users may submit more requests and thus impact the aggregated requests in cloud center.

We obtain the total utility obtained by user i ($i \in \mathcal{N}$) over all H future time slots as

$$\begin{aligned} U_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right) &= \sum_{h\in\mathcal{H}} U_i^h\left(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h\right) \\ &= \sum_{h\in\mathcal{H}} \left(w_i\,(b-r)\,\lambda_i^h - \delta^h \bar{T}_i^h\right), \end{aligned} \tag{16}$$

where $\boldsymbol{\lambda}_{-i} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_{i-1}, \boldsymbol{\lambda}_{i+1}, \ldots, \boldsymbol{\lambda}_n)$ denotes the $(n-1)$ $H \times 1$ vectors of all users' request profile except that of user i. In this paper, we assume that each user i ($i \in \mathcal{N}$) has a reservation value v_i. That is to say, cloud user i will prefer to use the cloud service if $U_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right) \geq v_i$ and refuse to use the cloud service otherwise.

For the cloud provider, its objective is trying to select an appropriate servers subset $\mathcal{S}$ from $\mathcal{M}$ and configure a proper request allocation strategy $\boldsymbol{p}_{\mathcal{S}}$, such that its net reward, i.e., the charge to all cloud users minus its energy cost, is maximized. We denote π as the net profit, then the cloud provider's problem is to maximize the value π. That is,

$$\begin{aligned} \text{maximize} \quad & \pi\left(\mathcal{S}, \boldsymbol{p}_{\mathcal{S}}\right) = r \sum_{i\in\mathcal{N}} \sum_{h\in\mathcal{H}} \lambda_i^h - H \sum_{j\in\mathcal{S}} E_j,\ \mathcal{S} \subseteq \mathcal{M}, \\ \text{s.t.} \quad & U_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right) \geq v_i,\ \boldsymbol{\lambda}_i \in \mathcal{Q}_i,\ \forall i \in \mathcal{N}, \\ & p_j^h \lambda_\Sigma^h < c_j \mu_j,\ \forall j \in \mathcal{S},\ \forall h \in \mathcal{H}, \\ & \textstyle\sum_{j\in\mathcal{S}} p_j^h = 1,\ \forall h \in \mathcal{H}. \end{aligned} \tag{17}$$

4 Game Formulation and Analyses

Since the multiple users have to compete for using the computing resources, and their strategies are subject to that of the cloud provider, we formulate the relationship between the cloud provider and its multiple users into a Stackelberg game. For the cloud provider, we try to approximate its server selection solution space by using a control parameter and configure an appropriate request allocation strategy to the selected servers. For the multiple users, we characterize their competitions as a non-cooperative game and formulate them into a generalized Nash equilibrium problem (GNEP). By employing variational inequality (VI) theory, we analyze the the formulated GNEP. Then, we propose an iterative algorithm (IA) to compute appropriate strategies for both the cloud provider and its multiple users.

4.1 Game Formulation

Game theory studies the problems in which multiple players try to maximize their utilities or minimize their disutilities. In this subsection, we characterize the optimization problem presented in Section 3.5 as a Stackelberg game, which is a sequential game played between a *Leader* and a set of *Followers* [36]. All of them try to maximize their own utilities.

In our work, the cloud provider plays the role of the leader, who tries to select an appropriate servers subset $\mathcal{S}$ from $\mathcal{M}$ and configure a proper request allocation strategy $\boldsymbol{p}_{\mathcal{S}}$ to the selected servers, such that it can appeal user requests as many as possible while its cost is relatively low. We denote $\mathcal{Q}_L$ as the servers selection space, then $\mathcal{Q}_L$ can be expressed as

$$\mathcal{Q}_L = \{\mathcal{S} | \mathcal{S} \subseteq \mathcal{M}\}. \tag{18}$$

Each cloud user is regarded as a follower, i.e., the set of followers is the n cloud users. Notice that when given $\mathcal{S}$ and $\boldsymbol{p}_{\mathcal{S}}$, the workload of each server j ($j \in \mathcal{S}$) in time slot h ($h \in \mathcal{H}$) never exceeds its processing capacity, i.e., $p_j^h \lambda_\Sigma^h < c_j \mu_j$ ($\forall j \in \mathcal{S}$). We denote σ as a relative small constant and add the constraint $\lambda_\Sigma^h \leq (1-\sigma)\,\lambda_{\text{up}}^h$, where $\lambda_{\text{up}}^h = \min_{j\in\mathcal{S}}\left\{c_j \mu_j / p_j^h\right\}$. Then the request strategy set of user i ($i \in \mathcal{N}$) can be expressed as

$$\hat{\mathcal{Q}}_i\left(\boldsymbol{\lambda}_{-i}\right) = \mathcal{Q}_i \cap \left\{ \boldsymbol{\lambda}_i \Big| \sum_{i=1}^{n} \lambda_i^h \leq (1-\sigma)\lambda_{\text{up}}^h,\ \forall h \in \mathcal{H} \right\}. \tag{19}$$

Then, the joint strategy set of all followers is given by $\hat{\mathcal{Q}}_F = \hat{\mathcal{Q}}_1 \times \cdots \times \hat{\mathcal{Q}}_n$.

A Stackelberg game assumes certain decision power for both the leader and followers, with the leader processing a higher priority. The followers have to make their decisions subject to the leader's strategy [37] and try to maximize their own utilities. Therefore, the profit maximization problem of the cloud provider can be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

formulated as the following optimization problem (OPT):

$$\text{maximize} \quad \pi(\mathcal{S}, \boldsymbol{p}_{\mathcal{S}}) = r \sum_{i \in \mathcal{N}} \sum_{h \in \mathcal{H}} \lambda_i^h - H \sum_{j \in \mathcal{S}} E_j, \ \mathcal{S} \in \mathcal{Q}_L,$$
$$\text{s.t.} \quad \boldsymbol{\lambda}_i \in \underset{\boldsymbol{\lambda}_i' \in \hat{\mathcal{Q}}_i(\boldsymbol{\lambda}_{-i})}{\arg\max} U_i\left(\boldsymbol{\lambda}_i', \boldsymbol{\lambda}_{-i}\right), \forall i \in \mathcal{N}. \tag{20}$$

Theorem 4.1. *Above OPT problem is NP-hard.*

Proof: A complete proof of the theorem is given in the supplementary material. □

4.2 Leader's Strategy Analysis

We now consider the problem from the perspective of the cloud provider. We try to reduce the servers selection space by using an approximated one. We also configure an optimal request allocation strategy for the aggregated requests.

4.2.1 Solution Space Approximation

To reduce the solution space ($\mathcal{Q}_L$), we use a parameter ε to categorize all the elements in $\mathcal{Q}_L$ and reduce some server subsets, which provide similar processing capacities.

We notice that a server j ($j \in \mathcal{M}$) gains net profit at most P_j in each time slot for the cloud provider, where $P_j = rc_j\mu_j - E_j$. Therefore, we use P_j to characterize the server j ($j \in \mathcal{M}$) during our approximation process. In this paper, the value P_j ($\forall j \in \mathcal{M}$) is assumed to be greater than zero. We denote $P_T(\mathcal{S})$ as the largest possible profit gained by all servers in $\mathcal{S}$. Then we have $P_T(\mathcal{S}) = \sum_{j \in \mathcal{S}} P_j$. Arrange all of the server subsets $\mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \ldots, \mathcal{S}^{(|\mathcal{Q}_L|)}$ in $\mathcal{Q}_L$, such that $P_T\left(\mathcal{S}^{(1)}\right) \leq P_T\left(\mathcal{S}^{(2)}\right) \leq \cdots \leq P_T\left(\mathcal{S}^{(|\mathcal{Q}_L|)}\right)$. We try to reduce some elements in $\mathcal{Q}_L$ to an approximated solution space $\mathcal{Q}_L^{(\varepsilon)}$, such that for each server subset $\mathcal{S}$ in $\mathcal{Q}_L$, there exists an element $\mathcal{S}^{(\varepsilon)}$ ($\mathcal{S}^{(\varepsilon)} \in \mathcal{Q}_L^{(\varepsilon)}$) satisfying $P_T(\mathcal{S}) \leq (1+\varepsilon) P_T\left(\mathcal{S}^{(\varepsilon)}\right)$. The idea is formalized in Algorithm 1.

Given $\varepsilon, r, \boldsymbol{\mu}, \boldsymbol{E}$, and $\mathcal{M}$, where $\boldsymbol{\mu} = (\mu_j)_{j \in \mathcal{M}}$ and $\boldsymbol{E} = (E_j)_{j \in \mathcal{M}}$, the algorithm $Calcualte\_\mathcal{Q}_L^{(\varepsilon)}$ finds an approximated solution space $\mathcal{Q}_L^{(\varepsilon)}$ for $\mathcal{Q}_L$. The key idea is trying to reduce solution space by selecting some server subset representatives and removing similar ones (Steps 7-11). At the beginning, we set $\mathcal{Q}_L^{(\varepsilon)}$ as $\{\emptyset\}$, which only contains an empty subset (Step 1). Then, for a server j ($j \in \mathcal{M}$), we merge it into each of the subsets in $\mathcal{Q}_L^{(\varepsilon)}$ (Steps 3-5) and resort the subsets according to their largest possible gained profits (Step 6). After resorting, we try to remove some later elements $\mathcal{S}^{(l)}$ ($\mathcal{S}^{(l)} \in \mathcal{Q}_L$) if there exists a previous one $\mathcal{S}^{(p)}$ ($\mathcal{S}^{(p)} \in \mathcal{Q}_L$) satisfying $P_T\left(\mathcal{S}^{(l)}\right) \leq (1+\varepsilon) P_T\left(\mathcal{S}^{(p)}\right)$ (Steps 7-11). This process is terminated when all the servers in $\mathcal{M}$ are considered.

Theorem 4.2. *The time complexity of Algorithm 1 is* $\Theta\left(m\mathcal{L}^{(\varepsilon)} \log \mathcal{L}^{(\varepsilon)}\right)$, *where*

$$\mathcal{L}^{(\varepsilon)} = \frac{(1+\varepsilon) \ln P_T(\mathcal{M})}{\varepsilon}, \tag{21}$$

Algorithm 1 $Calculate\_\mathcal{Q}_L^{(\varepsilon)}(\varepsilon, r, \boldsymbol{\mu}, \boldsymbol{E}, \mathcal{M})$

Input: $\varepsilon, r, \boldsymbol{\mu}, \boldsymbol{E}, \mathcal{M}$.
Output: $\mathcal{Q}_L^{(\varepsilon)}$.
1: *Initialization*: For each server j ($j \in \mathcal{M}$), calculate P_j. Set $\mathcal{Q}_L^{(\varepsilon)} = \{\emptyset\}$.
2: **for** (each server $j \in \mathcal{M}$) **do**
3: **for** (each element $\mathcal{S} \in \mathcal{Q}_L^{(\varepsilon)}$) **do**
4: Merge server j into set $\mathcal{S}$, i.e., set $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$.
5: **end for**
6: Sort the elements in $\mathcal{Q}_L^{(\varepsilon)}$ such that $P_T\left(\mathcal{S}^{(1)}\right) \leq P_T\left(\mathcal{S}^{(2)}\right) \leq \ldots \leq P_T\left(\mathcal{S}^{(|\mathcal{Q}_L^{(\varepsilon)}|)}\right)$.
7: **for** (i from 1 to $\left|\mathcal{Q}_L^{(\varepsilon)}\right| - 1$) **do**
8: **if** ($P_T\left(\mathcal{S}^{(i+1)}\right) \leq (1+\varepsilon) P_T\left(\mathcal{S}^{(i)}\right)$) **then**
9: Remove $\mathcal{S}^{(i+1)}$ from $\mathcal{Q}_L^{(\varepsilon)}$, i.e., set $\mathcal{Q}_L^{(\varepsilon)} \leftarrow \mathcal{Q}_L^{(\varepsilon)} - \left\{\mathcal{S}^{(i+1)}\right\}$.
10: **end if**
11: **end for**
12: **end for**
13: **return** $\mathcal{Q}_L^{(\varepsilon)}$.

with $P_T(\mathcal{M}) = \sum_{j \in \mathcal{M}} (rc_j\mu_j - E_j)$.

Proof: We first derive an upper bound of the length of set $\mathcal{Q}_L^{(\varepsilon)}$. As can be seen from Algorithm 1, after steps 7-11, two continuous elements $\mathcal{S}$ and $\mathcal{S}'$ ($\mathcal{S}, \mathcal{S}' \in \mathcal{Q}_L^{(\varepsilon)}$) satisfy the condition $P_T\left(\mathcal{S}'\right) \Big/ P_T(\mathcal{S}) > 1+\varepsilon$. That is to say, for any two elements $\mathcal{S}, \mathcal{S}' \in \mathcal{Q}_L^{(\varepsilon)}$, $P_T\left(\mathcal{S}'\right) \Big/ P_T(\mathcal{S}) > 1+\varepsilon$. Therefore, $\mathcal{Q}_L^{(\varepsilon)}$ contains element $\emptyset$ and may contain at most other $\left\lceil \log_{1+\varepsilon} P_T(\mathcal{M}) \right\rceil$ elements. Thus, the number of elements in set $\mathcal{Q}_L^{(\varepsilon)}$ is at most

$$\begin{aligned} \log_{(1+\varepsilon)} P_T(\mathcal{M}) + 1 &= \frac{\ln P_T(\mathcal{M})}{\ln(1+\varepsilon)} + 1 \\ &\leq \frac{(1+\varepsilon) \ln P_T(\mathcal{M})}{\varepsilon} + 1. \end{aligned}$$

In Algorithm 1, we note that the for loop (Steps 3-5) requires $\Theta\left(\mathcal{L}^{(\varepsilon)}\right)$ to complete as well as the other for loop (Steps 7-11). In step 6, it takes at most $\Theta\left(L^{(\varepsilon)} \log L^{(\varepsilon)}\right)$ to sort the elements in set $\mathcal{Q}_L^{(\varepsilon)}$. Therefore, the outer for loop (Steps 2-12) takes time $\Theta\left(m\left(2\mathcal{L}^{(\varepsilon)} + \mathcal{L}^{(\varepsilon)} \log \mathcal{L}^{(\varepsilon)}\right)\right) = \Theta\left(m\mathcal{L}^{(\varepsilon)} \log \mathcal{L}^{(\varepsilon)}\right)$. Thus, the time complexity of Algorithm 1 is $\Theta\left(m + m\mathcal{L}^{(\varepsilon)} \log \mathcal{L}^{(\varepsilon)}\right) = \Theta\left(m\mathcal{L}^{(\varepsilon)} \log \mathcal{L}^{(\varepsilon)}\right)$. This completes the proof and the result follows. □

4.2.2 Request Distribution Analysis

After a servers subset is determined, the cloud provider has to consider an appropriate request allocation strategy, such that the average response time over all users (see Eq. (14)) is minimized and thus satisfies and appeals more cloud users. Before address the request allocation strategy, we first show two properties which are presented in Theorem 4.3 and Corollary 4.4.

Theorem 4.3. *Consider a two server system such that* $\mu_j < \mu_k$ $(j, k \in \mathcal{M})$, *it is optimal to assign a certain amount of*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

requests λ ($\lambda < \lambda_\Sigma$ and $\lambda < \mu_j + \mu_k$) to both of the servers. Namely, the minimum $\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)$ occurs at $0 < x < 1$, where

$$\tilde{T}_l(xp_c) = \left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\left(\frac{(xp_c)^2}{\mu_l} + \frac{(xp_c)^2}{c_l\mu_l - xp_c\lambda_\Sigma}\right), \tag{22}$$

$l = j, k$, and $p_c = \lambda/\lambda_\Sigma$.

Proof: From (22), we can observe that $\tilde{T}_k(xp_c)$ and $\tilde{T}_j((1-x)p_c)$ are convex and thus $\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)$ is a convex function. We try to obtain the minimum of $\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)$ by analyzing its derivative for all $0 \le x \le 1$. After some algebraic calculation, we obtain

$$\frac{d}{dx}\left(\tilde{T}_k(xp_c)\right) = \left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\frac{2xp_c^2}{\mu_k} + \left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\frac{2xp_c^2c_k\mu_k - (x)^2p_c^3\lambda_\Sigma}{(c_k\mu_k - xp_c\lambda_\Sigma)^2},$$

and

$$\frac{d}{dx}\left(\tilde{T}_j((1-x)p_c)\right) = -\left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\frac{2(1-x)p_c^2}{\mu_j} - \left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\frac{2(1-x)p_c^2c_j\mu_j - (1-x)^2p_c^3\lambda_\Sigma}{(c_j\mu_j - (1-x)p_c\lambda_\Sigma)^2}.$$

Obviously, $\frac{d}{dx}\left(\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)\right) < 0$ at $x = 0$, and $\frac{d}{dx}\left(\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)\right) > 0$ at $x = 1$. Besides, we can further obtain

$$\frac{d^2}{dx^2}\left(\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)\right) = \left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\left(\frac{2p_c^2}{\mu_k} + \frac{2p_c^2c_k^2\mu_k^2}{(c_k\mu_k - xp_c\lambda_\Sigma)^3}\right) + \left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\left(\frac{2p_c^2}{\mu_j} + \frac{2p_c^2c_j^2\mu_j^2}{(c_j\mu_j - (1-x)p_c\lambda_\Sigma)^3}\right) > 0.$$

Therefore, $\frac{d}{dx}\left(\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)\right)$ is strictly increasing over $0 \le x \le 1$ so long as $x\lambda < \mu_k$ and $(1-x)\lambda < \mu_j$, that is, for all feasible assignments.

Thus, the minimum $\tilde{T}_k(xp_c) + \tilde{T}_j((1-x)p_c)$ occurs at $0 < x < 1$ and the result follows. □

Corollary 4.4. *Given a server set $\mathcal{S}$, it is optimal for a cloud provider to assign jobs to all servers in $\mathcal{S}$.*

Proof: We denote $T_\mathcal{R}(\boldsymbol{p}_\mathcal{R})$ as the average response time of all users when the server set is $\mathcal{R}$ ($\mathcal{R} \subset \mathcal{S}$) and the request distribution strategy is $\boldsymbol{p}_\mathcal{R}$, where $\boldsymbol{p}_\mathcal{R} = (p_j)_{j\in\mathcal{R}}$. Denote $\boldsymbol{p}^*_\mathcal{R}$ as an optimal assignment. Let

$$\tilde{T}_j(p_j) = \left(\frac{1}{\lambda_\Sigma}\sum_{i\in\mathcal{N}}\lambda_i^2\right)\left(\frac{p_j^2}{\mu_j} + \frac{p_j^2}{c_j\mu_j - xp_j\lambda_\Sigma}\right),$$

for all $j \in \mathcal{R}$. Then we have

$$T_\mathcal{R}(\boldsymbol{p}_\mathcal{R}) = \sum_{j\in\mathcal{R}}\tilde{T}_j(p_j).$$

Assuming that there exist a server k ($k \in \mathcal{R}$) and a server l ($l \in \bar{\mathcal{R}}$), where $\bar{\mathcal{R}}$ denotes the supplementary set of $\mathcal{R}$. Based on Theorem 4.3, we can find that there exists an x ($0 < x < 1$) such that

$$\tilde{T}_k(xp_k^*) + \tilde{T}_l((1-x)p_k^*) < \tilde{T}_k(p_k^*).$$

Namely,

$$\sum_{j\in\mathcal{R},j\neq k}\tilde{T}_j(p_j^*) + \tilde{T}_k(xp_k^*) + \tilde{T}_l((1-x)p_k^*) < \sum_{j\in\mathcal{R}}\tilde{T}_j(p_j^*).$$

Therefore, there exists a probability vector $\boldsymbol{p}'_{\mathcal{R}\cup\{l\}}$ for server set $\mathcal{R}\cup\{l\}$ with $p'_k = xp_k^*$, $p'_l = (1-x)p_k^*$, and $p'_j = p_j^*$ ($j \in \mathcal{R}$ and $j \neq k$), such that

$$T_{\mathcal{R}\cup\{l\}}\left(\boldsymbol{p}'_{\mathcal{R}\cup\{l\}}\right) < T_\mathcal{R}(\boldsymbol{p}^*_\mathcal{R}).$$

This process can be terminated when the server set $\bar{\mathcal{R}}$ is empty. Thus, it is optimal for a cloud provider to assign requests to all servers for set $\mathcal{S}$, and the result follows. □

Next, we focus on the probability distribution for the minimization of system response time in time slot h ($h \in \mathcal{H}$) (see Eq. (14)). We denote P as the constraint of probability, i.e.,

$$P = \sum_{j\in\mathcal{S}}p_j^h = 1, \tag{23}$$

and try to minimize $\bar{T}^h$ by using the method of Lagrange multiplier, namely,

$$\frac{\partial\bar{T}^h}{\partial p_j^h} = \phi\frac{\partial P}{\partial p_j^h} = \phi, \tag{24}$$

where ϕ is a Lagrange multiplier. That is,

$$\frac{\partial\bar{T}^h}{\partial p_j^h} = \left(\sum_{i\in\mathcal{N}}\frac{\left(\lambda_i^h\right)^2}{\lambda_\Sigma^h}\right)\left(\frac{2p_j^h}{\mu_j} + \frac{2p_j^hc_j\mu_j - \left(p_j^h\right)^2\lambda_\Sigma^h}{\left(c_j\mu_j - p_j^h\lambda_\Sigma^h\right)^2}\right) = \phi, \tag{25}$$

for all $j \in \mathcal{S}$, and $\sum_{j\in\mathcal{S}}p_j^h = 1$.

Since the second order of $\bar{T}^h\left(p_j^h\right)$ is

$$\frac{\partial^2\bar{T}^h}{\partial\left(p_j^h\right)^2} = \left(\sum_{i\in\mathcal{N}}\frac{\left(\lambda_i^h\right)^2}{\lambda_\Sigma^h}\right)\left(\frac{2}{\mu_j} + \frac{2c_j^2\mu_j^2}{\left(c_j\mu_j - p_j^h\lambda_\Sigma^h\right)^3}\right) > 0, \tag{26}$$

we can conclude that $\frac{\partial\bar{T}^h}{\partial p_j^h}$ is an increasing positive function on p_j^h. Based on above derivations, we propose an algorithm to calculate $\boldsymbol{p}_\mathcal{S}$, which is motivated by [9].

Given $\epsilon, \boldsymbol{\mu}, \lambda_\Sigma^h$, and $\mathcal{S}$, our optimal request allocation algorithm to find $\boldsymbol{p}_\mathcal{S}$ is given in Algorithm 2. The algorithm uses another subalgorithm $Calculate\_p_j^h$, which, given μ_j, λ_Σ^h, and ϕ, finds p_j^h satisfies (25). The key observation is that the left-hand side of (25) is an increasing

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

function on p_j^h (see (26)). Therefore, given ϕ, we can find p_j^h by using the binary search method in certain interval $[lb, ub]$ (Steps 2-9 in Algorithm 3). We set lb simply as 0. For ub, we may note that the allocated load never exceed its processing capacity. Therefore, we set ub as $\min\left\{1, c_j\mu_j/\lambda_\Sigma^h\right\}$. The value of ϕ can also be found by using the binary search method (Steps 2-8 in Algorithm 2). The search interval $[lb, ub]$ for ϕ is determined as follows. We set lb simply as 0. As for ub, we set an increment variable inc, which is initialized as a relative small positive constant and repeatedly doubled (Step (7)). The value of inc is added to ϕ to increase ϕ until the sum of p_j^h ($j \in \mathcal{S}$) found by $Calculate\_p_j^h$ is at least 1 (Steps 2-8). Once $[lb, ub]$ is decided, ϕ can be searched by using binary search (Steps 10-20). After ϕ is determined (Step 21), p_j^h can be computed (Steps 22-24).

Algorithm 2 $Calculate\_\boldsymbol{p}_{\mathcal{S}}^h(\epsilon, \boldsymbol{\mu}, \lambda_\Sigma^h, \mathcal{S})$

Input: $\epsilon, \boldsymbol{\mu}, \lambda_\Sigma^h, \mathcal{S}$
Output: $\boldsymbol{p}_{\mathcal{S}}^h$.
1: *Initialization*: Let inc be a relative small positive constant. Set $\boldsymbol{p}_{\mathcal{S}}^h \leftarrow \mathbf{0}$, and $\phi \leftarrow 0$.
2: **while** $(\sum_{j\in\mathcal{S}} p_j^h < 1)$ **do**
3: Set $mid \leftarrow \phi + inc$, and $\phi \leftarrow mid$.
4: **for** (each server $j \in \mathcal{S}$) **do**
5: $p_j^h \leftarrow Calculate\_p_j^h(\epsilon, \mu_j, \lambda_\Sigma^h, \phi)$.
6: **end for**
7: Set $inc \leftarrow 2 \times inc$.
8: **end while**
9: Set $lb \leftarrow 0$ and $ub \leftarrow \phi$.
10: **while** $(ub - lb > \epsilon)$ **do**
11: Set $mid \leftarrow (ub + lb)/2$, and $\phi \leftarrow mid$.
12: **for** (each server $j \in \mathcal{S}$) **do**
13: $p_j^h \leftarrow Calculate\_p_j^h(\epsilon, \mu_j, \lambda_\Sigma^h, \phi)$.
14: **if** $(\sum_{j\in\mathcal{S}} p_j^h < 1)$ **then**
15: Set $lb \leftarrow mid$.
16: **else**
17: Set $ub \leftarrow mid$.
18: **end if**
19: **end for**
20: **end while**
21: Set $\phi \leftarrow (ub + lb)/2$.
22: **for** (each server $j \in \mathcal{S}$) **do**
23: $p_j^h \leftarrow Calculate\_p_j^h(\epsilon, \mu_j, \lambda_\Sigma^h, \phi)$.
24: **end for**
25: **return** $\boldsymbol{p}_{\mathcal{S}}^h$.

By Algorithm 3, we note that the while loop (Steps 2-9) is a binary search process, which is very efficient and requires time $\Theta\left(\log\left(\frac{1}{\epsilon}\min\left\{1, \frac{\mu_j}{\lambda_\Sigma^h}\right\}\right)\right) = \Theta\left(\log\left(\frac{1}{\epsilon}\right)\right)$, where ϵ is the error tolerance (e.g., 0.1, 0.01, in our work, ϵ is set as 0.01). As for Algorithm 2, its main idea is the twice uses of binary search method. Specifically, the first while loop (Steps 2-8) is the first use of binary search method, which is designed to determine an upper bound (ub) of ϕ. The second use of binary search method is the second while loop (Steps 10-20), which is designed to search the exact ϕ such that $\sum_{j\in\mathcal{S}} p_j^h = 1$. Therefore, the number of loops of the first while loop is $\Theta(\log ub)$ and that of the second while loop is $\Theta\left(\log\left(\frac{ub}{\epsilon}\right)\right)$. To analyze the time complexity of Algorithm 2, we have to find an

Algorithm 3 $Calculate\_p_j^h(\epsilon, \mu_j, \lambda_\Sigma^h, \phi)$

Input: $\epsilon, \mu_j, \lambda_\Sigma^h, \phi$.
Output: p_j^h.
1: *Initialization*: Set $ub \leftarrow \min\left\{1, \frac{\mu_j}{\lambda_\Sigma^h}\right\}$, and $lb \leftarrow 0$.
2: **while** $(ub - lb > \epsilon)$ **do**
3: Set $mid \leftarrow (ub + lb)/2$, and $p_j^h \leftarrow mid$.
4: **if** $(\frac{\partial}{\partial p_j^h}\bar{T}^h(p_j^h) < \phi)$ **then**
5: Set $lb \leftarrow mid$.
6: **else**
7: Set $ub \leftarrow mid$.
8: **end if**
9: **end while**
10: Set $p_j^h \leftarrow (ub + lb)/2$.
11: **return** p_j.

upper bound of ϕ, i.e., ub. From (25) of our paper, we know that

$$\begin{aligned}
\phi = \frac{\partial \bar{T}^h}{\partial p_j^h} &= \left(\sum_{i\in\mathcal{N}} \frac{\left(\lambda_i^h\right)^2}{\lambda_\Sigma^h}\right)\left(\frac{2p_j^h}{\mu_j} + \frac{2p_j^h c_j\mu_j - \left(p_j^h\right)^2\lambda_\Sigma^h}{\left(c_j\mu_j - p_j^h\lambda_\Sigma^h\right)^2}\right) \\
&\le \left(\sum_{i\in\mathcal{N}} \lambda_i^h\right)\left(\frac{2p_j^h}{\mu_j} + \frac{2p_j^h c_j\mu_j - \left(p_j^h\right)^2\lambda_\Sigma^h}{\left(c_j\mu_j - p_j^h\lambda_\Sigma^h\right)^2}\right) \\
&= \frac{2p_j^h\lambda_\Sigma^h}{\mu_j} + \frac{2c_j\mu_j\left(p_j^h\lambda_\Sigma^h\right) - \left(p_j^h\lambda_\Sigma^h\right)^2}{\left(c_j\mu_j - p_j^h\lambda_\Sigma^h\right)^2} \\
&< 2c_j + \frac{2c_j\mu_j\left(p_j^h\lambda_\Sigma^h\right)}{\left(c_j\mu_j - p_j^h\lambda_\Sigma^h\right)^2} \\
&\le 2c_j + \frac{2\left(1-\sigma\right)\left(c_j\mu_j\right)^2}{\sigma^2\left(c_j\mu_j\right)^2} \\
&\le 2c_{\max} + \frac{2\left(1-\sigma\right)}{\sigma^2},
\end{aligned} \tag{27}$$

where $c_{\max}$ denotes the maximal number of cores of a server, i.e., $c_{\max} = \max_{j\in\mathcal{M}}(c_j)$, and σ is a relative small positive constant (to maintain the convexity of the individual strategy set of a user, $p_j^h\lambda_\Sigma^h \le (1-\sigma)\,c_j\mu_j$ ($\forall j \in \mathcal{M}$) is added to the optimization problem of the cloud provider as a constraint, i.e., we try to maximize the profits under the constraint $p_j^h\lambda_\Sigma^h \le (1-\sigma)\,c_j\mu_j$ ($\forall j \in \mathcal{M}$). In our work, σ is also set as 0.01). Therefore, an upper bound of ϕ, i.e., ub, in Algorithm 2 is $\left(2c_{\max} + \frac{2(1-\sigma)}{\sigma^2}\right)$. We can conclude that the number of loops of the first while loop (Steps 2-8) is $\Theta\left(\log\left(c_{\max} + \frac{(1-\sigma)}{\sigma^2}\right)\right)$, and the number of loops of the second while loop (Steps 10-20) is $\Theta\left(\log\left(\frac{c_{\max}}{\epsilon} + \frac{(1-\sigma)}{\epsilon\sigma^2}\right)\right)$. Since at each iteration (loop) of these two while loops, the main operation is the $|\mathcal{S}|$ times call for Algorithm 3, and the possible maximal value of $|\mathcal{S}|$ is m, these two while loops require time $\Theta\left(m\log\left(\frac{1}{\epsilon}\right)\log\left(c_{\max} + \frac{(1-\sigma)}{\sigma^2}\right)\right)$ to complete. In addition, the time complexity of the for loop (Steps 22-24) requires time $\Theta\left(m\log\left(\frac{1}{\epsilon}\right)\right)$. Therefore, the time complexity of Algorithm 2 is $\Theta\left(m\log\left(\frac{1}{\epsilon}\right)\log\left(\frac{c_{\max}}{\epsilon} + \frac{(1-\sigma)}{\epsilon\sigma^2}\right)\right)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

4.3 Followers' Decisions Analysis

We formulate the competitions among multiple users as a generalized Nash equilibrium problem (GNEP). By employing variational inequality (VI) theory, we analyze the existence of a generalized Nash equilibrium solution set. And then we propose an algorithm to compute a generalized Nash equilibrium (GNE).

4.3.1 GNEP Formulation

We formulate the profit optimization problem of all the cloud users as a generalized Nash equilibrium problem (GNEP), in which each user selfishly optimizes his/her own profit within his strategy set that also depends on the strategies of the other users [38].

As mentioned earlier, all users are considered to be selfish and each user i ($i \in \mathcal{N}$) tries to maximize his/her utility or minimize his/her disutility, while ignoring those of others and satisfying the global constraint. In view of (20), we can observe that user i's ($i \in \mathcal{N}$) optimization problem is equivalent to

$$\begin{aligned} \text{minimize} \quad & f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right) = \sum_{h \in \mathcal{H}} \left(\delta^h T_i^h - w_i\left(r-c\right)\lambda_i^h\right), \\ \text{s.t.} \quad & \left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right) \in \hat{\mathcal{Q}}_F. \end{aligned} \tag{28}$$

The above formulation GNEP can be formally defined by the tuple $G = \left\langle \hat{\mathcal{Q}}_F, \boldsymbol{f} \right\rangle$, where $\boldsymbol{f} = (f_1, \ldots, f_n)$. The aim of user i ($i \in \mathcal{N}$), given the other users' strategies $\boldsymbol{\lambda}_{-i}$, is to choose an $\boldsymbol{\lambda}_i \in \hat{\mathcal{Q}}_i\left(\boldsymbol{\lambda}_{-i}\right)$ such that his/her disutility function $f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right)$ is minimized.

Definition 4.1. *A generalized Nash equilibrium (GNE) of the game $G = \left\langle \hat{\mathcal{Q}}_F, \boldsymbol{f} \right\rangle$ is a strategy profile $\boldsymbol{\lambda}^*$ such that for each user i ($i \in \mathcal{N}$):*

$$\boldsymbol{\lambda}_i^* \in \underset{\lambda_i \in \hat{\mathcal{Q}}_i\left(\boldsymbol{\lambda}_{-i}^*\right)}{\arg\min} f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}^*\right), \boldsymbol{\lambda}^* \in \hat{\mathcal{Q}}_F. \tag{29}$$

At the generalized Nash equilibrium, each user cannot further decrease its disutility by choosing a different strategy while the strategies of other users are fixed. The equilibrium strategy profile can be found when each user's strategy is the best response to the strategies of other users.

4.3.2 GNE Existence Analysis

We try to analyze the existence of generalized Nash equilibrium for the formulated GNEP by employing variational inequality (VI) theory. When passing from the GNEP (see Eq. (28)) to the associated VI, the solutions of the GNEP that are also solutions of VI are termed as variational solutions [39], and enjoy some remarkable properties that make them particularly appealing in many applications. Before address the problem, we show two properties which are presented in Theorem 4.5 and Theorem 4.6.

Theorem 4.5. *For each cloud user i ($i \in \mathcal{N}$), the set $\mathcal{Q}_i$ is closed and convex, and each disutility function $f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right)$ is continuously differentiable in $\boldsymbol{\lambda}_i$. For each fixed tuple $\boldsymbol{\lambda}_{-i}$, the disutility function $f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right)$ is convex in $\boldsymbol{\lambda}_i$ over the set Ω_i.*

Proof: A complete proof of the theorem is given in the supplementary material. □

Theorem 4.6. *Every solution of the variational inequality (VI) problem, denoted by $\mathrm{VI}\left(\hat{\mathcal{Q}}_F, \mathbf{F}\right)$, is a solution of the GNEP $G = \left\langle \hat{\mathcal{Q}}_F, \boldsymbol{f} \right\rangle$, where*

$$\mathbf{F}\left(\boldsymbol{\lambda}\right) = \left(\mathbf{F}_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right)\right)_{i=1}^n, \tag{30}$$

with

$$\mathbf{F}_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right) = \nabla_{\boldsymbol{\lambda}_i} f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{-i}\right). \tag{31}$$

Proof: A complete proof of the theorem is given in the supplementary material. □

Theorem 4.7. *If $\lambda_\Sigma^h \leq \min_{j \in \mathcal{S}} \left\{\left(n+1\right) c_j \mu_j / \left(2n p_j^h\right)\right\}$ ($h \in \mathcal{H}$), there exists a generalized Nash equilibrium solution set for the formulated GNEP $G = \left\langle \hat{\mathcal{Q}}_F, \boldsymbol{f} \right\rangle$.*

Proof: A complete proof of the theorem is given in the supplementary material. □

4.3.3 GNE Computation

With the establishment of the generalized Nash equilibrium (GNE) of the GNEP G, we now aim at obtaining a suitable algorithm to compute the GNE.

Note that we can further rewrite the optimization problem (28) as follows:

$$\begin{aligned} \text{minimize} \quad & f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_\Sigma\right) = \sum_{h \in \mathcal{H}} \left(\delta^h \bar{T}_i^h - w_i\left(r-c\right)\lambda_i^h\right), \\ \text{s.t.} \quad & \boldsymbol{\lambda}_i \in \hat{\mathcal{Q}}_i. \end{aligned} \tag{32}$$

with

$$\bar{T}_i^h\left(\lambda_i^h, \lambda_\Sigma^h\right) = \sum_{j \in \mathcal{S}} \left(\frac{\left(p_j^h\right)^2 \lambda_i^h}{\mu_j} + \frac{\left(p_j^h\right)^2 \lambda_i^h}{c_j \mu_j - p_j^h \lambda_\Sigma^h} \right), \tag{33}$$

where $\boldsymbol{\lambda}_\Sigma$ denotes the aggregated request profile of all users over the H future time slots, i.e., $\boldsymbol{\lambda}_\Sigma = \sum_{i=1}^n \boldsymbol{\lambda}_i$. From the above equation, we can see that the calculation of the disutility function of each individual user only requires the knowledge of the aggregated request profile of all users ($\boldsymbol{\lambda}_\Sigma$) rather than the specific individual request profile of all other users ($\boldsymbol{\lambda}_{-i}$), which can bring two advantages. On the one hand, it can reduce communication traffic between users and the cloud provider. On the other hand, it can also keep privacy for each individual user to certain extent, which is seriously considered by many cloud users.

We can compute the variational solutions of the GNEP (20) by solving the following Nash equilbirium (NEP). This can be done by using the framework in [39], which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

Algorithm 4 $Calculate\_\boldsymbol{\lambda}(\epsilon, \mathcal{S}, \boldsymbol{p}_{\mathcal{S}}, \tau)$

Input: $\epsilon, \mathcal{S}, \boldsymbol{p}_{\mathcal{S}}, \tau$.
Output: $\boldsymbol{\lambda}$.

Initialization: Randomly choose a feasible strategy vector $\boldsymbol{z}^{(0)} = \left(\boldsymbol{\lambda}^{(0)}, \boldsymbol{\eta}^{(0)}\right)$ $(\boldsymbol{z}^{(0)} \in \mathcal{Q}_{F,\eta})$. Set $\bar{\boldsymbol{\lambda}} \leftarrow \mathbf{0}$, $\bar{\boldsymbol{\eta}} \leftarrow 0$, and $k \leftarrow 0$.
while $\left(\left\|\boldsymbol{z}^{(k)} - \boldsymbol{z}^{(k-1)}\right\| > \epsilon\right)$ **do**
  for (each cloud user $i \in \mathcal{N}$) **do**
    Receive $\boldsymbol{\lambda}_{\Sigma}^{(k)}$ from the cloud provider and compute $\boldsymbol{\lambda}_i^{(k+1)}$ as follows:

$$\boldsymbol{\lambda}_i^{(k+1)} \in \underset{\boldsymbol{\lambda}_i \in \mathcal{Q}_i}{\arg\min} \left\{ \begin{array}{c} f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}^{(k)}\right) + \boldsymbol{\eta}^{T(k)} \boldsymbol{\psi}\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}^{(k)}\right) \\ + \frac{\tau}{2}\left\|\boldsymbol{\lambda}_i - \bar{\boldsymbol{\lambda}}_i\right\|^2 \end{array} \right\}.$$

    Send the updated strategy to the cloud provider.
  end for
  The cloud provider computes $\eta^{(k+1)}$ as

$$\boldsymbol{\eta}^{(k+1)} \in \underset{\boldsymbol{\eta} \geq \mathbf{0}}{\arg\min} \left\{ -\boldsymbol{\eta}^T \boldsymbol{\psi}\left(\boldsymbol{\lambda}\right) + \frac{\tau}{2}\|\boldsymbol{\eta} - \bar{\boldsymbol{\eta}}\|^2 \right\}.$$

  if (Nash equilibrium is reached) **then**
    The $n+1$ cloud users updates their centroids: $\left(\bar{\boldsymbol{\lambda}}, \bar{\boldsymbol{\eta}}\right) \leftarrow \left(\boldsymbol{\lambda}^{(k+1)}, \boldsymbol{\eta}^{(k+1)}\right)$
  end if
  Set $k \leftarrow k+1$.
end while
return $\boldsymbol{\lambda}^{(k)}$.

leads to an algorithm $Calculate\_\boldsymbol{\lambda}$. Specifically, the $n+1$ users try to solve the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}\right) + \boldsymbol{\eta}^T \boldsymbol{\psi}\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}\right), \\ \text{s.t.} \quad & \boldsymbol{\lambda}_i \in \mathcal{Q}_i, \ \forall i \in \mathcal{N}, \\ \text{minimize} \quad & -\boldsymbol{\eta}^T \boldsymbol{\psi}\left(\boldsymbol{\lambda}\right), \\ \text{s.t.} \quad & \boldsymbol{\eta} \geq \mathbf{0}. \end{aligned} \tag{34}$$

where $\boldsymbol{\eta} = \left(\eta_h\right)_{h=1}^{H}$, and

$$\boldsymbol{\psi}\left(\boldsymbol{\lambda}\right) = \left(\sum_{i=1}^{n} \lambda_i^h - \Gamma_h\right)_{h=1}^{H}, \tag{35}$$

with

$$\Gamma_h = \min_{j \in \mathcal{S}} \left\{ \left(n+1\right) c_j \mu_j / \left(2 n p_j^h\right) \right\}, \ h \in \mathcal{H}. \tag{36}$$

That is to say, when given the aggregated requests, we must find a strategy vector $\boldsymbol{z}^* = (\boldsymbol{\lambda}^*, \boldsymbol{\eta}^*) \in \mathcal{Q}_{F,\eta}$, where $\mathcal{Q}_{F,\eta} = Q_F \times \mathbb{R}_+^{(H)}$, such that

$$\boldsymbol{\lambda}_i^* \in \underset{\boldsymbol{\lambda}_i \in \mathcal{Q}_i}{\arg\min} \left\{ f_i\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}^*\right) + \boldsymbol{\eta}^T \boldsymbol{\psi}\left(\boldsymbol{\lambda}_i, \boldsymbol{\lambda}_{\Sigma}^*\right) + \frac{\tau}{2}\left\|\boldsymbol{\lambda}_i - \bar{\boldsymbol{\lambda}}_i\right\|^2 \right\}, \tag{37}$$

for each user i $(i \in \mathcal{N})$, and

$$\boldsymbol{\eta}^* \in \underset{\boldsymbol{\eta} \geq \mathbf{0}}{\arg\min} \left\{ -\boldsymbol{\eta}^T \boldsymbol{\psi}\left(\boldsymbol{\lambda}^*\right) + \frac{\tau}{2}\|\boldsymbol{\eta} - \bar{\boldsymbol{\eta}}\|^2 \right\}. \tag{38}$$

where τ $(\tau > 0)$ is a regularization parameter and can guarantee the convergence of the algorithm $Calculate\_\boldsymbol{\lambda}$ if its value is large enough [39]. The idea is formalized in Algorithm 4.

4.4 An Iterative Algorithm

In this section, we describe the whole process of our proposed service mechanism, which is formalized in Algorithm 5.

We describe operational process of the proposed iterative algorithm. At the beginning, the cloud provider approximates its sever selection space $(\mathcal{Q}_L)$ and obtains the approximated one $(\mathcal{Q}_L^{(\varepsilon)})$. For each servers subset $(\tilde{\mathcal{S}})$ in $(\mathcal{Q}_L^{(\varepsilon)})$, it initializes the allocation strategy $(\boldsymbol{p}_{\tilde{\mathcal{S}}})$ in different time slot h $(h \in \mathcal{H})$. Under this servers subset and allocation strategy, all of the users calculate the proper request strategies. The cloud provider reconfigures the allocation strategy such that the average response time over all users is minimized. Each of the user in the current set $(\mathcal{S}_c)$ calculates its utility, if the value is less than its reserved value (v_i), then he/she refuses to use the cloud service. This process is terminated when all of the users who choose the cloud service and their corresponding request strategies are kept unchanged. The algorithm terminates until it selects the optimal servers subset from the approximated subset solution space $(\mathcal{Q}_L^{(\varepsilon)})$.

Algorithm 5 Iterative Algorithm (IA)

Input: $\varepsilon, \boldsymbol{\mu}, a, b, r, \tau, \mathcal{M}$
Output: $\mathcal{S}, \boldsymbol{p}_{\mathcal{S}}$.

Initialization: The cloud provider approximates its solution space, i.e., $\mathcal{Q}_L^{(\varepsilon)} \leftarrow Calculate\_\mathcal{Q}_L^{(\varepsilon)}\left(\varepsilon, c, \boldsymbol{\mu}, \boldsymbol{E}, \mathcal{M}\right)$. Set $\pi_{\mathcal{S}} \leftarrow 0$.
for (each server subset $\tilde{\mathcal{S}} \in \mathcal{Q}_L^{(\varepsilon)}$) **do**
  Set $\mathcal{S}_c \leftarrow \mathcal{N}$, and $\mathcal{S}_l \leftarrow \emptyset$.
  for (each time slot $h \in \mathcal{H}$) **do**
    for (each server $j \in \tilde{\mathcal{S}}$) **do**
      Set $p_j^h = \mu_j / \left(\sum_{j \in \tilde{\mathcal{S}}} \mu_j\right)$.
    end for
  end for
  while $(\mathcal{S}_c \neq \mathcal{S}_l)$ **do**
    Set $\mathcal{S}_l \leftarrow \mathcal{S}_c$, and $\boldsymbol{\lambda} \leftarrow Calculate\_\boldsymbol{\lambda}\left(\varepsilon, \mathcal{S}, \boldsymbol{p}_{\mathcal{S}}, \tau\right)$.
    for (each time slot $h \in \mathcal{H}$) **do**
      Set $\boldsymbol{p}_{\tilde{\mathcal{S}}}^h \leftarrow Calculate\_\boldsymbol{p}_{\tilde{\mathcal{S}}}{}^h\left(\varepsilon, \boldsymbol{\mu}, \lambda_{\Sigma}^h, \mathcal{S}\right)$.
    end for
    for (each user $i \in \mathcal{S}_c$) **do**
      if $\left(U_i\left(\boldsymbol{\lambda}_i^{(k)}, \boldsymbol{\lambda}_{\Sigma}^{(k)}\right) < v_i\right)$ **then**
        Set $\boldsymbol{\lambda}_i \leftarrow \mathbf{0}$, and $\mathcal{S}_c \leftarrow \mathcal{S}_c - \{i\}$.
      end if
    end for
  end while
  Set $\pi_{\tilde{\mathcal{S}}} \leftarrow c \sum_{i \in \mathcal{N}} \sum_{h \in \mathcal{H}} \lambda_i^h - E_T\left(\tilde{\mathcal{S}}\right)$.
  if $(\pi_{\tilde{\mathcal{S}}} > \pi_{\mathcal{S}})$ **then**
    Set $\pi_{\mathcal{S}} \leftarrow \pi_{\tilde{\mathcal{S}}}$, $\mathcal{S} \leftarrow \tilde{\mathcal{S}}$, and $\boldsymbol{p}_{\mathcal{S}} \leftarrow \boldsymbol{p}_{\tilde{\mathcal{S}}}$.
  end if
end for
return $\mathcal{S}, \boldsymbol{p}_{\mathcal{S}}$.

5 Performance Evaluation

In this section, we provide some numerical results to validate our theoretical analyses and illustrate the performance of our proposed IA algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

TABLE 2: System parameters

System parameters	(Fixed)–[Varied range] (increment)
Servers set control parameter (ε)	(0.2)–[0.2, 1.0] (0.2)
Number of cloud users (n)	(50)–[5, 50] (5)
Energy parameters (ξ_j, a_j)	[0.01, 2.5], 3
Weight value (w_i)	[1, 10]
User total requests (Λ_i)	35
Reservation value (v_i)	0
Other parameters ($b, m, \mu_{\mathcal{M}}, r, c, \delta$)	(0.02, 50, 800, 100, 60, 1.1)

In the following simulation results, we assume that the number of cloud users is at most 50 over future H time slots, which is not a very long period of time. Specifically, each time slot is set as one hour of a day and H is set as 24. As shown in Table 2, the server set controlling parameter (ε) is varied from 0.2 to 1.0 with increment 0.2. The number of cloud users (n) is varied from 5 to 50 with increment 5. For each server j ($j \in \mathcal{M}$), the energy consumption parameter ξ_j is randomly chosen from 0.01 to 2.5 and a_j is set as a constant 3. Each cloud user i ($i \in \mathcal{N}$) chooses a weight value from 1 to 10 to balance his/her net profit and time utility. For simplicity, the reservation value v_i and total requests Λ_i for each user i ($i \in \mathcal{N}$) are set as 0 and 35, respectively. Market benefit factor r is set to 100, per request charge by the cloud provider c is equal to 60, and δ is set as 1.1. The cost of one unit of energy is set as 0.02. In our simulation, the number of servers (m) in the cloud provider is set as 50 and its total processing capacity ($\mu_{\mathcal{M}}$) is equal to 800.

5.1 Results of One Instance

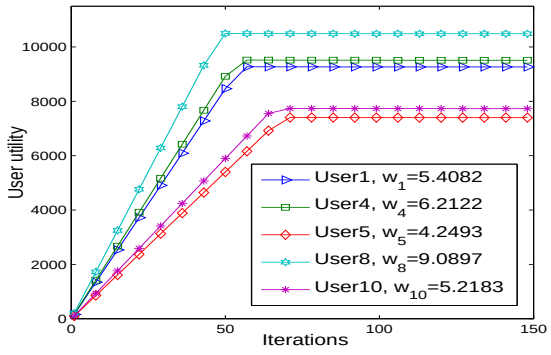


Fig. 2: Convergence process

Fig. 2 presents the utility results for five different cloud users versus the number of iterations of our proposed $Calculate\_\boldsymbol{\lambda}$ algorithm (Algorithm 4) in a certain instance. Specifically, it presents the utility results of 5 randomly selected cloud users (users 1, 4, 5, 8, and 10). We can observe that the utilities of all users seem to linearly increase and finally reach a relatively stable state with the increase of iteration number. The reason behind lies in that the request strategies of all users are kept unchanged, i.e., reach a generalized Nash equilibrium solution after some iterations. In addition, the utility with a larger weight value reaches a relatively stable state more faster. This trend also reflects the convergence process of our proposed IA algorithm at each iteration. It can be seen that the utility of each user has already achieved a relatively stable state after about 80 iterations, which reflects the high efficiency of the developed algorithm.

In Fig. 3, we plot the request profile of some cloud users for a scenario of 50 users. Specifically, it presents the requests shape of some users over future 24 time slots. We randomly select 3 users (users 25, 38, and 42). It can be seen that the requests of all users tend to decrease with the delay of time slot. The reason behind lies in the fact that in our proposed model, we take into average response time into account and the deteriorating factor grows exponentially, which also demonstrates the downward trend of the aggregated requests shown in Fig. 4, i.e., the aggregated requests slightly decrease with the delay of time slot.

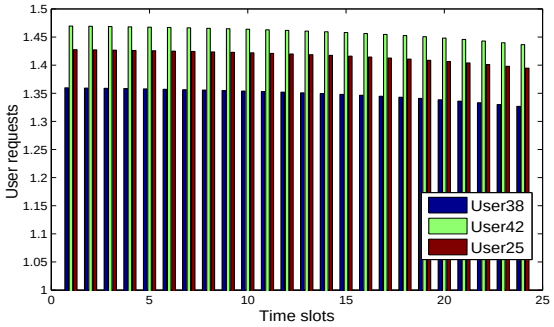


Fig. 3: Specific user requests

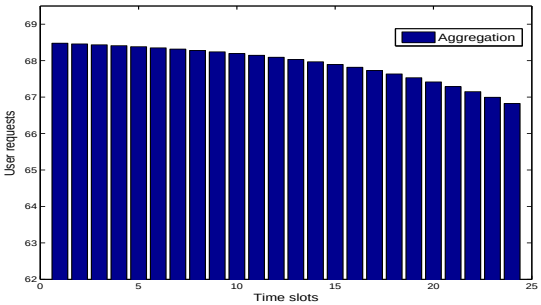


Fig. 4: Aggregated load

In Fig. 5, we present the impacts of different servers subset. Table 3 shows an instance of servers subset when ε is 0.2. In that table, we show the first 8 server subset obtained by our $Calculate\_\mathcal{Q}_L^{(\varepsilon)}$ algorithm (Algorithm 1). Fig. 5 shows the corresponding results. Specifically, it shows the total charge C_T from all users, where $C_T = c\sum_{i\in\mathcal{N}}\sum_{h\in\mathcal{H}}\lambda_i^h$, total energy cost E_T, where $E_T = H\sum_{j\in\mathcal{S}}E_j$, and net profit $\pi = C_T - E_T$ over future H time slots. As can be seen from Fig. 5, at first, the net profit of the cloud provider increases with the increase of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2017.2701793, IEEE Transactions on Cloud Computing

TABLE 3: System parameters

No. of subset	Servers to provide service
1	{50}
2	{49, 50}
3	{48, 49, 50}
4	{47, 48, 49, 50}
5	{44, 45, 46, 47, 48, 49, 50}
6	{41, 42, 43, 44, 45, 46, 47, 48, 49, 50}
7	{31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50}
8	{27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50}

total processing capacity of provided servers. However, it decreases after the number of subset exceeds 4. The reason behind lies in the fact that at the beginning, the aggregated requests from all users can not exceed the total processing capacity provided by the cloud provider (i.e., $\lambda_{\Sigma}^{h} < \mu_{\mathcal{S}}, \forall h \in \mathcal{H}$), while the provided processing capacity is large enough, the aggregated requests can not rise more due to their individual limits (i.e., $\lambda_i^h < \Lambda_i, \forall i \in \mathcal{N}$). This is also the reason that the total charge (C_T) increases at first and reaches a relatively stable state when the processing capacity is large enough, as well as the trend of energy cost and thus reflects the results of net profit (see Fig. 5).

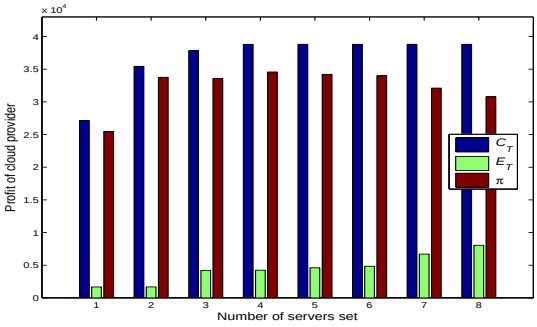


Fig. 5: Impact of servers

5.2 Results of Various Configuration Instances

To simulate the heterogeneous system and the different preferences of multiple cloud users, i.e., the different preferences over payments and time efficiencies, we randomly generate the server parameter (ξ_j) for each server and the weight value (w_i) for each user according to Table 2. For the simulated results, we perform 300 runs, of which the average value is computed.

Fig. 6 and Fig. 7 show the impacts of the number of cloud users and the value of ε. In Fig. 6, we compare the net profit (π) obtained by our IA algorithm with that of using all 50 servers (π_T). The number of cloud users increases from 5 to 50 with increment 5. As mentioned above, we perform 300 runs and compute the average value. As shown in Fig. 6, we also present the maximal and minimal profit values over the 300 runs.

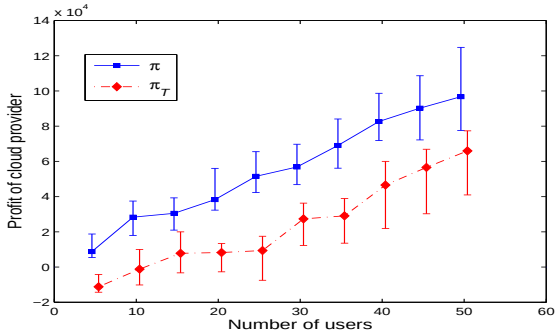


Fig. 6: Impact of users

Obviously, the average net profit value obtained by our IA algorithm increases with the increase of the number of cloud users. We can also observe that the net profit by using all servers is negative at the beginning. The reason behind lies in that the aggregated requests from all users are not enough while the total energy cost of all servers is large. However, our results are always better than those of by using all servers. This shows that our IA algorithm can select appropriate servers to provide services. Fig. 7 shows the impact of ε. It can be seen that the average net profit value obtained by IA algorithm is the largest when ε is set to 0.2. The reason behind lies in the fact that the smaller the value of ε is, it takes more probability for our algorithm to select an appropriate servers subset equalling to the optimal one, that is, it takes more probability that the optimal servers subset is included in our approximated solution space.

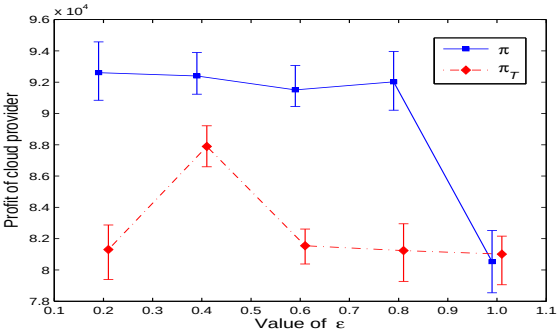


Fig. 7: Impact of ε

6 Conclusions and Future Work

With the popularization of cloud computing and its many advantages such as cost-effectiveness, elasticity, and scalability, more and more applications are moved from local computing environment to cloud center. In this work, we try to design a new service mechanism for profit optimizations of both a cloud provider and its multiple users.

We consider the problem from a game theoretic perspective and characterize the relationship between the cloud provider and its multiple users as a Stackelberg

game, in which the strategies of all users are subject to that of the cloud provider. The cloud provider tries to select appropriate servers and configure a proper request allocation strategy to reduce energy cost while satisfying its cloud users at the same time. We approximate its server selection space by adding a controlling parameter and configure an optimal request allocation strategy. For each user, we design a utility function which combines the net profit with time efficiency and try to maximize its value under the strategy of the cloud provider. We formulate the competitions among all users as a generalized Nash equilibrium problem (GNEP). We solve the problem by employing varational inequality (VI) theory and prove that there exists a generalized Nash equilibrium solution set for the formulated GNEP. Finally, we propose an iterative algorithm (IA), which characterizes the whole process of our proposed service mechanism. We conduct some numerical calculations to verify our theoretical analyses. The experimental results show that our IA algorithm can reduce energy cost and improve users utilities to certain extent by configuring proper strategies.

As part of future work, we will study the cloud center choice among multiple different cloud providers or determine a proper mixed choice strategy. Another direction is the opposite, we consider problem from cloud providers and study the competitions among multiple cloud providers, which may incorporate charge price, service quality, and so on.

ACKNOWLEDGMENTS

We are very grateful to the associate editor and anonymous reviewers for their comments and suggestions which have significantly improved the quality of the manuscript. The research was partially funded by the Key Program of National Natural Science Foundation of China (Grant No. 61432005), the National Outstanding Youth Science Program of National Natural Science Foundation of China (Grant No. 61625202), the International (Regional) Cooperation and Exchange Program of National Natural Science Foundation of China (Grant No. 61661146006), the National Natural Science Foundation of China (Grant Nos. 61370095, 61472124, 61602170), the International Science & Technology Cooperation Program of China (Grant Nos. 2015DFA11240), the National Key R&D Program of China (Grant No. 2016YFB0201402), and the Chinese Postdoctoral Science Foundation (Grant Nos. 2016M602409, 2016M602410).

REFERENCES

[1] A. Prasad and S. Rao, "A mechanism design approach to resource procurement in cloud computing," *Computers, IEEE Transactions on*, vol. 63, no. 1, pp. 17–30, Jan 2014.

[2] R. Pal and P. Hui, "Economic models for cloud service markets: Pricing and capacity planning," *Theoretical Computer Science*, vol. 496, no. 0, pp. 113 – 124, 2013.

[3] P. D. Kaur and I. Chana, "A resource elasticity framework for qos-aware execution of cloud applications," *Future Generation Computer Systems*, vol. 37, no. 0, pp. 14 – 25, 2014.

[4] L. Duan, D. Zhan, and J. Hohnerlein, "Optimizing cloud data center energy efficiency via dynamic prediction of cpu idle intervals," in *2015 IEEE 8th International Conference on Cloud Computing*. IEEE, 2015, pp. 985–988.

[5] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, "Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds," *IEEE Transactions on Services Computing*, 2015, doi: 10.1109/TSC.2015.2466545.

[6] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM computer communication review*, vol. 39, no. 1, pp. 68–73, 2008.

[7] J. Cao, K. Hwang, K. Li, and A. Zomaya, "Optimal multiserver configuration for profit maximization in cloud computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1087–1096, June 2013.

[8] Y. Feng, B. Li, and B. Li, "Price competition in an oligopoly market with multiple iaas cloud providers," *Computers, IEEE Transactions on*, vol. 63, no. 1, pp. 59–73, Jan 2014.

[9] J. Cao, K. Li, and I. Stojmenovic, "Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers," *Computers, IEEE Transactions on*, vol. 63, no. 1, pp. 45–58, Jan 2014.

[10] S. Jrgensen and G. Zaccour, "A survey of game-theoretic models of cooperative advertising," *European Journal of Operational Research*, vol. 237, no. 1, pp. 1 – 14, 2014.

[11] S. Liu, S. Ren, G. Quan, M. Zhao, and S. Ren, "Profit aware load balancing for distributed cloud data centers," in *Parallel Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, May 2013, pp. 611–622.

[12] U. Lampe, M. Siebenhaar, A. Papageorgiou, D. Schuller, and R. Steinmetz, "Maximizing cloud provider profit from equilibrium price auctions," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, June 2012, pp. 83–90.

[13] H. Goudarzi and M. Pedram, "Maximizing profit in cloud computing system via resource allocation," in *Proceedings of the 2011 31st International Conference on Distributed Computing Systems Workshops*, ser. ICDCSW '11. IEEE Computer Society, 2011, pp. 1–6.

[14] E. Körpeoğlu, A. Şen, and K. Güler, "Non-cooperative joint replenishment under asymmetric information," *European Journal of Operational Research*, vol. 227, no. 3, pp. 434–443, 2013.

[15] C. Liu, K. Li, C. Xu, and K. Li, "Strategy configurations of multiple users competition for cloud service reservation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 508–520, 2016.

[16] M. J. Osborne and A. Rubinstein, "A course in game theory," *MIT press*, 1994.

[17] S. S. Aote and M. U. Kharat, "A game-theoretic model for dynamic load balancing in distributed systems," in *Proceedings of the International Conference on Advances in Computing, Communication and Control*, ser. ICAC3 '09. ACM, 2009, pp. 235–238.

[18] N. Li and J. Marden, "Designing games for distributed optimization," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 7, no. 2, pp. 230–242, April 2013.

[19] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 4, pp. 537 – 555, 2011.

[20] G. Scutari and J.-S. Pang, "Joint sensing and power allocation in nonconvex cognitive radio games: Nash equilibria and distributed algorithms," *Information Theory, IEEE Transactions on*, vol. 59, no. 7, pp. 4626–4661, July 2013.

[21] Z. Wang, A. Szolnoki, and M. Perc, "Rewarding evolutionary fitness with links between populations promotes cooperation," *Journal of Theoretical Biology*, vol. 349, no. 0, pp. 50 – 56, 2014.

[22] G. Scutari, D. Palomar, F. Facchinei, and J.-S. Pang, "Convex optimization, game theory, and variational inequality theory," *Signal Processing Magazine, IEEE*, vol. 27, no. 3, pp. 35–49, May 2010.

[23] K. Li, C. Liu, and K. Li, "An approximation algorithm based on game theory for scheduling simple linear deteriorating jobs," *Theoretical Computer Science*, vol. 543, no. 0, pp. 46 – 51, 2014.

[24] C. A. Ioannou and J. Romero, "A generalized approach to belief learning in repeated games," *Games and Economic Behavior*, vol. 87, no. 0, pp. 178 – 203, 2014.

[25] K. Li, C. Liu, K. Li, and A. Y. Zomaya, "A framework of price bidding configurations for resource usage in cloud computing,"

IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 8, pp. 2168–2181, 2016.

[26] P. Wang, Y. Qi, and X. Liu, "Power-aware optimization for heterogeneous multi-tier clusters," *Journal of Parallel and Distributed Computing*, vol. 74, no. 1, pp. 2005 – 2015, 2014.

[27] Y. Gao, H. Guan, Z. Qi, T. Song, F. Huan, and L. Liu, "Service level agreement based energy-efficient resource management in cloud data centers," *Computers & Electrical Engineering*, vol. 40, no. 5, pp. 1621 – 1633, 2014.

[28] J. Mei, K. Li, J. Hu, S. Yin, and E. H.-M. Sha, "Energy-aware preemptive scheduling algorithm for sporadic tasks on {DVS} platform," *Microprocessors and Microsystems*, vol. 37, no. 1, pp. 99 – 112, 2013.

[29] D. Zhu, R. Melhem, and B. Childers, "Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 14, no. 7, pp. 686–700, July 2003.

[30] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755 – 768, 2012, special Section: Energy efficiency in large-scale distributed systems.

[31] R. N. Calheiros, A. N. Toosi, C. Vecchiola, and R. Buyya, "A coordinator for scaling elastic applications across multiple clouds," *Future Generation Computer Systems*, vol. 28, no. 8, pp. 1350 – 1362, 2012, including Special sections SS: Trusting Software Behavior and SS: Economics of Computing Services.

[32] M. Mezmaz, N. Melab, Y. Kessaci, Y. Lee, E.-G. Talbi, A. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing*, vol. 71, no. 11, pp. 1497 – 1508, 2011.

[33] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," *Design Automation Conference*, vol. 0, pp. 868–873, 2004.

[34] I. Atzeni, L. Ordonez, G. Scutari, D. Palomar, and J. Fonollosa, "Noncooperative day-ahead bidding strategies for demand-side expected cost minimization with real-time adjustments: A gnep approach," *Signal Processing, IEEE Transactions on*, vol. 62, no. 9, pp. 2397–2412, 2014.

[35] H. Chen, Y. Li, R. Louie, and B. Vucetic, "Autonomous demand side management based on energy consumption scheduling and instantaneous load billing: An aggregative game approach," *Smart Grid, IEEE Transactions on*, vol. 5, no. 4, pp. 1744–1754, 2014.

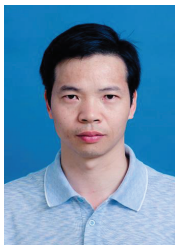
[36] H. Soliman and A. Leon-Garcia, "Game-theoretic demand-side management with storage devices for the future smart grid," *Smart Grid, IEEE Transactions on*, vol. 5, no. 3, pp. 1475–1485, 2014.

[37] G. Du, R. J. Jiao, and M. Chen, "Joint optimization of product family configuration and scaling design by stackelberg game," *European Journal of Operational Research*, vol. 232, no. 2, pp. 330 – 341, 2014.

[38] J. Wang, M. Peng, S. Jin, and C. Zhao, "A generalized nash equilibrium approach for robust cognitive radio networks via generalized variational inequalities," *Wireless Communications, IEEE Transactions on*, vol. 13, no. 7, pp. 3701–3714, 2014.

[39] G. Scutari, D. Palomar, F. Facchinei, and J.-S. Pang, "Monotone games for cognitive radio systems," in *Distributed Decision Making and Control*, ser. Lecture Notes in Control and Information Sciences. Springer London, 2012, vol. 417, pp. 83–112.

Chubo Liu received the BS and PhD degrees in computer science and technology from Hunan University, China, in 2011 and 2016, respectively. His research interests include mainly in modeling and scheduling of distributed computing systems, approximation and randomized algorithms, game theory, grid, and cloud computing. He has published several papers in journals such as the *IEEE Transactions on Parallel and Distributed Systems*, the *Future Generation Computer Systems*, and *the Theoretical Computer Science*.



Kenli Li received the PhD degree in computer science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar with the University of Illinois at Urbana-Champaign from 2004 to 2005. He is currently the dean and a full professor of computer science and technology with Hunan University and deputy director of National Supercomputing Center in Changsha. His major research areas include parallel computing, high-performance computing, grid, and cloud computing. He has published more than 150 research papers in international conferences and journals such as the *IEEE Transactions on Computers*, the *IEEE Transactions on Parallel and Distributed Systems*, the *IEEE Transactions on Signal Processing*, the *Journal of Parallel and Distributed Computing*, ICPP, and CCGrid. He serves on the editorial board of the *IEEE Transactions on Computers*. He is an outstanding member of the CCF. He is a senior member of the IEEE.

Keqin Li is a SUNY Distinguished Professor of computer science. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU-GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber-physical systems. He has published over 480 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Sustainable Computing*. He is an IEEE Fellow.

Rajkumar Buyya is a Professor of Computer Science and Software Engineering; Future Fellow of the Australian Research Council; and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercialising its innovations in Grid and Cloud Computing. He received B.E and M.E in Computer Science and Engineering from Mysore and Bangalore Universities in 1992 and 1995 respectively; and a Doctor of Philosophy (PhD) in Computer Science and Software Engineering from Monash University, Melbourne, Australia in 2002. He was the founding Editor-in-Chief (EiC) of IEEE Transactions on Cloud Computing (TCC). He has authored over 500 publications and five text books including "Mastering Cloud Computing" published by McGraw Hill, Morgan Kaufmann, and China Machine Press for Indian, International, and Chinese markets respectively.