An approximation algorithm based on Non-cooperative game Theory for a Cloud Provider and Its Users

[1] N.Mounika, [2]P.Neelima.
[1]PG Student, Dept. of CSE, School of Engineering & Technology, Sri Padmavati Mahila University (Women's University), Tirupati .
[2]Assistant Professor, (Ph.D), Dept. of CSE, Sri Padmavati Mahila University (Women's University), Tirupati .

Abstract— In this paper, we endeavor to design an accommodation mechanism for profit optimizations of both a cloud provider and its multiple users. We consider the quandary 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 endeavors to cull and provision opportune servers and configure an opportune request allocation strategy to reduce energy cost while satiating its cloud users concurrently. We approximate its server's cull space by integrating a controlling parameter and configure an optimal request allocation strategy. For each utilizer, we design a utility function which amalgamates the net profit with time efficiency and endeavor to maximize its value under the strategy of the cloud provider. We formulate the competitions among all users as a generalized Nash equilibrium quandary (GNEP). We solve the quandary by employing variational inequality (VI) theory and prove that there subsists a generalized Nash equilibrium solution set for the formulated GNEP. Conclusively, we propose an iterative algorithm (IA), which characterizes the whole process of our proposed accommodation 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 opportune strategies.

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

I INTRODUCTION

C Loud computing is an increasingly popular paradigm of offering subscription-oriented accommodations to enterprises and consumers [1]. Conventionally, the provided accommodations refer to Infrastructure as a Accommodation (IaaS), Platform as a Accommodation (PaaS), and Software as a Accommodation (SaaS), which are all made available to the general public in a pay-as-you-go manner [2], [3]. To fortify sundry accommodations, 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 utilized for powering and colling the physical servers [5]. There are withal 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 cognate categories [6]. Hence, it is consequential to reduce energy cost for amending the profit of a cloud provider. However, it can often be visually perceived 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 accommodation quality. Consequently, it is paramount for a cloud provider to cull felicitous servers to provide accommodations, such that it reduces cost as much as possible while satiating its users concurrently.

For a cloud provider, the income (i.e., the revenue) is the accommodation charge to the aggregated requests from all cloud users [7]. When the per request charge is tenacious, servers cull and request allocation strategy are two consequential 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 utilize cloud accommodation and thus withal impact the profit. Concretely, if the provided computing capacity is

immensely colossal enough (i.e., many servers are under-utilized), this will result in tremendous amount of energy waste with immensely colossal cost and thus reduces the profit of the cloud provider. On the other hand, if the cloud provider provides less computing capacity or infelicitously configures the request allocation strategy, this will lead to low accommodation quality (e.g, long task replication time) and thus dissatisfies its cloud users or potential cloud users in the market.

The rest of the paper is organized as follows. Section II describes the models of the system and presents the quandary to be solved. Section III formulates the quandary into a Stackelberg game, which consists of a bellwether and a set of adherents. We analyze the strategies for both of the bellwether and the adherents. Many analyses and several sub algorithms are presented in this section. Section IV is developed to verify our theoretical analysis and show the efficacy of our proposed algorithm. We conclude the paper with future work in Section V.

II 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 M = {1, 2, m}. Each server j (j ∈ M) consists of cj cores and similar to [9], it is modeled by an M/M/c queueing system. We denote the set of cloud users as N = {1, 2 . . . 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).

A. 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 S from M (i.e., S ⊆ M) to provide services for the H future time slots, and configure a proper strategy pS = (p 1 S , . . . , p H S) with p h S = (p h j) j∈S (h ∈ 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 ∈ N) is equipped with a utility function (Ui) and a request configuration strategy (λi), i.e., the request strategy over H future time slots. All requests enter a queue to be processed by the cloud center. Let λΣ be the aggregated request vector, then we have λΣ = ∑ i∈N λi . The cloud provider tries to select an appropriate servers subset S, configure an appropriate allocation strategy pS , and publishes some information (e.g., per request charge r, server subset S, and the corresponding allocation strategy pS , current aggregated requests λΣ) 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. 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.

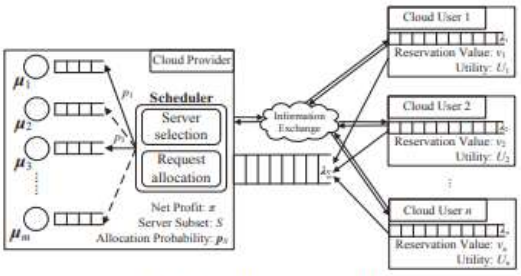

Fig. 1: Architecture model

B 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

$$E = dCV^2 f, \quad (1)$$

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, i.e., μ ∝ f. With reference to [9], [33], we also obtain f ∝ V ϕ with 0 < ϕ ≤ 1, which implies that V ∝ f 1/ϕ . Therefore, we know that the energy consumption is E ∝ f a and E ∝ μ a , where a = 1 + 2/ϕ ≥ 3. In this paper, we assume that

$$E = \xi \mu^a, \quad (2)$$

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

$$E_j = \chi c_j \xi_j \mu_j^{a_j}, \quad (3)$$

where μj is the processing rate of one core of server j, ξj and aj are the corresponding energy consumption factors.

C Request Profile Model

We consider a user request model similar to [34], [35], where the user i ′ s (i ∈ N) request profile over the H future time slots is formulated as

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

where λ h i (h ∈ H) is the arrival rate of requests from user i in the hth time slot and it is subject to the constraint 0 ≤ λ h i ≤ Λi , where Λi 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

$$\mathcal{Q}_i = \left\{ \boldsymbol{\lambda}_i | 0 \le \lambda_i^h \le \Lambda_i, \forall h \in \mathcal{H} \right\}, \quad (5)$$

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

D Cloud Service Model

The cloud provider is equipped with a request scheduler and m heterogeneous multicore servers. Each server j (j ∈ M) consists of cj 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 ∈ M) is presented by its service rate μj . Energy consumption factors ξj and aj are also different among different servers. The cloud provider only selects a servers subset S (S ⊆ M) to provide services.

Let p h j be the probability that each of the requests is assigned to server j (j ∈ S) in time slot h (h ∈ H) and ρ h j be the corresponding service utilization. Then we have $\rho_j^h = p_j^h \lambda_\Sigma^h / (c_j \mu_j)$, where λ h Σ denotes the aggregated requests from all cloud users in time slot h, i.e., λ h Σ = ∑n i=1 λ h i . Let π h k,j 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

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

where

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

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

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

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

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

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

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

where P h q,j 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 h q,j (∀j ∈ S) is assumed to be 1, and we have

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

With a request rate of λ h i (i ∈ N) in time slot h (h ∈ H), the average response time of user i on server j (j ∈ S) is given by

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

We derive the mean response time of user i (i ∈ N) over all servers as

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

and the average response time over all users as

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

D Problem Formulation

Now, let us consider user i ′ s (i ∈ 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. Hence, in this paper, we assume that the deteriorating rate of time utility is δ (δ > 1). Denote the Tˆh i as the time utility of user i in time slot h. Then we have Tˆh i = δ hT¯h i . More formally, the utility of user i (i ∈ N) in time slot h is defined as

$$U_i^h(\lambda_i^h, \boldsymbol{\lambda}_{-i}^h) = w_i R_i^h - \hat{T}_i^h$$
$$= w_i(b-r)\lambda_i^h - \delta^h \bar{T}_i^h, \quad (15)$$

where $\boldsymbol{\lambda}_{-i}^h = (\lambda_1^h, \ldots, \lambda_{i-1}^h, \lambda_i^h, \ldots, \lambda_n^h)$ denotes the vector of all users′ request profile in time slot h except that of user i, and wi (wi > 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 ∈ N) over all H future time slots as

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

where λ−i = (λ1, . . . , λi−1, λi+1, . . . , λn) denotes the (n − 1) H × 1 vectors of all users′ request profile except that of user i. In this paper, we assume that each user i (i ∈ N) has a reservation value vi . That is to say, cloud user i will prefer to use the cloud service if Ui (λi , λ−i) ≥ vi and refuse to use the cloud service otherwise. For the cloud provider, its objective is trying to select an appropriate servers subset S from M and configure a proper request allocation strategy pS , 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,

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

IV 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.

A 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 S from M and configure a proper request allocation strategy pS to the selected servers, such that it can appeal user requests as many as possible while its cost is relatively low. We denote QL as the servers selection space, then QL can be expressed as

$$\mathcal{Q}_L = \{\mathcal{S} | \mathcal{S} \subseteq \mathcal{M}\}. \quad (18)$$

Each cloud user is regarded as a follower, i.e., the set of followers is the n cloud users. Notice that when given S and pS , the workload of each server j (j ∈ S) in time slot h (h ∈ H) never exceeds its processing capacity, i.e., p h j λ h Σ < cjμj (∀j ∈ S). We denote σ as a relative small constant and add the constraint λ h Σ ≤ (1 − σ) λ h up, where λ h up = minj∈S { cjμj / p h j } . Then the request strategy set of user i (i ∈ N) can be expressed as

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

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 formulated as the following optimization problem (OPT):

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

V PERFORMANCE EVALUATION

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

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 ∈ M), the energy consumption parameter ξj is randomly chosen from 0.01 to 2.5 and aj is set as a constant 3. Each cloud user i (i ∈ N) chooses a weight value from 1 to 10 to balance his/her net profit and time utility. For simplicity, the reservation value vi and total requests Λi for each user i (i ∈ 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 (μM) is equal to 800.

A. Results of One Instance

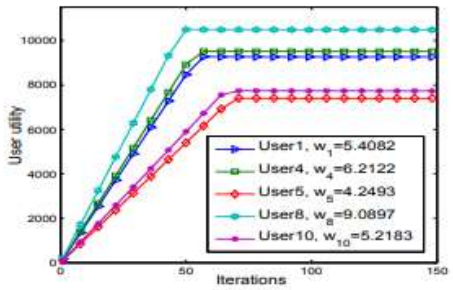

Fig. 2: Convergence process

Fig. 2 presents the utility results for five different cloud users versus the number of iterations of our proposed Calculate λ 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.

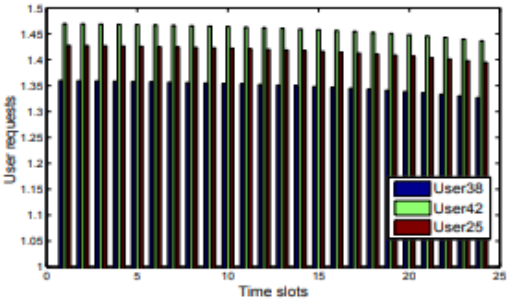

Fig. 3: Specific user requests

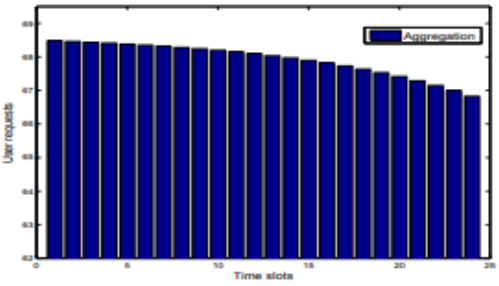

Fig. 4: Aggregated load

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 Q (ε) L algorithm (Algorithm 1). Fig. 5 shows the corresponding results. Specifically, it shows the total charge CT from all users, where CT = c ∑ i∈N ∑ h∈H λ h i , total energy cost ET , where ET = H ∑ j∈S Ej , and net profit π = CT −ET 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 oftotal 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., λ h Σ < μS , ∀h ∈ H), while the provided processing capacity is large enough, the aggregated requests can not rise more due to their individual limits (i.e., λ h i < Λi , ∀i ∈ N). This is also the reason that the total charge (CT) 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).

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}

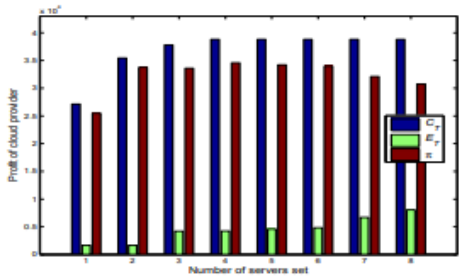

Fig. 5: Impact of servers

B Results of Various Configurations

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 (ξj) for each server and the weight value (wi) 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 (πT). 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.

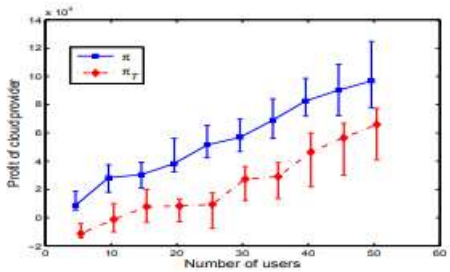

Fig. 6: Impact of users

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.

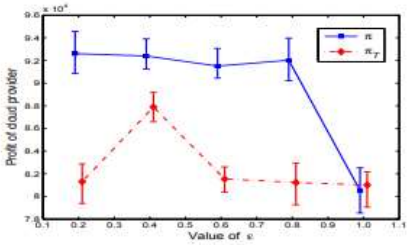

Fig. 7: Impact of ε

VI 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.

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. Korpeo ¨ glu, A. S¸en, and K. G ˇ uler, "Non-cooperative joint re- ¨ plenishment 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,

[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.

N.Mounika was born in AP, India. Currently she is studying her Post graduate degree in School of Engineering & Technology, Sri Padamavathi Mahila Vishwa Vidhyalayam, Tirupathi in Department of Computer Science & Engineering.

P.Neelima is currently working as an Assistant Professor in CSE department, School of Engineering & Technology, Sri Padamavathi Mahila Vishwa Vidhyalayam,Tirupati.