Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 915878, 14 pages
http://dx.doi.org/10.1155/2014/915878

Research Article

A Game Theory Approach to Fair and Efficient Resource Allocation in Cloud Computing

Xin Xu and Huiqun Yu

Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China

Correspondence should be addressed to Huiqun Yu; yhq@ecust.edu.cn

Received 5 November 2013; Revised 14 January 2014; Accepted 14 January 2014; Published 24 April 2014

Academic Editor: Balaji Raghavan

On-demand resource management is a key characteristic of cloud computing. Cloud providers should support the computational resource sharing in a fair way to ensure that no user gets much better resources than others. Another goal is to improve the resource utilization by minimizing the resource fragmentation when mapping virtual machines to physical servers. The focus of this paper is the proposal of a game theoretic resources allocation algorithm that considers the fairness among users and the resources utilization for both. The experiments with an FUGA implementation on an 8-node server cluster show the optimality of this algorithm in keeping fairness by comparing with the evaluation of the Hadoop scheduler. The simulations based on Google workload trace demonstrate that the algorithm is able to reduce resource wastage and achieve a better resource utilization rate than other allocation mechanisms.

1. Introduction

Cloud computing is a new paradigm that provides computational resources as a highly scalable service in a pay-as-you-go model and implements high performance computing in a distributed way [1]. Virtualization technologies in clouds enable the on-demand resource allocation by creating different types of VMs on physical servers [2]. A set of possible VM types are typically defined by cloud providers and each VM type consists of different quantities of resources (CPU, memory, disk storage, etc.). It is up to cloud providers to make intelligent decisions on how to allocate the heterogeneous resources of physical servers to those required VMs.

First of all, the fairness problem is considered in resource allocation, which means no one is allocated much better resources than others. In multiresource environment, resources of various types, such as CPU, memory, and disk storage, are required by users with different demands. In this scenario, fair allocation aims to equalize the largest resource fraction of total availability allocated to each user [3].

The other goal of resource allocation is to guarantee the computational resources to be fully utilized. Due to the variety of resource requirements with different VM types, lots of resource fragments in physical servers could be generated during the VM deployment. Therefore, an efficient resource allocation method should minimize the amount of resources fragments.

Motivated by these goals, the allocation problem considered in our work is based on two key principles. One is that multiple types of resources should be shared among users in fair way. The other is that complementary types of VMs are packed on physical servers in order to better utilize the underlying resources.

This paper proposes a resource allocation algorithm based on game theory for multiresource environment. The problem is modeled as a finite extensive game. Each physical server providing resources is treated as a game player and knows the utility information of other players. To achieve a fair allocation among users while keeping a high resource utilization level, we design a fairness-utilization tradeoff utility function. A measurement is established for fairness based on the dominant resource fairness (DRF) mechanism [3]. We also focus on two key issues for improving resource utilization, that is, (1) trying to maximize the minimum consumption among these multiple resources, (2) lowering the uneven consumption of different resources.

In summary, the main contributions of this paper are as follows.

(i) A cloud resource management system is designed to provide on-demand resources in time.

(ii) The multiresource allocation problem on virtual machine level is modeled as a finite extensive game with perfect information and the utility function is designed by trading off fairness and resource utilization.

(iii) A game theoretic resource allocation algorithm is proposed to get an optimal resource allocation decision, which guarantees fairness of multiple resources sharing among separated users and reduces the resource fragments to increase the efficiency.

This paper is organized as follows. In Section 2, the background and related work are presented. In Section 3, we present the architecture of the cloud resource management system and the mathematical model of resource allocation. Section 4 proposes a game theoretic resource allocation algorithm FUGA. The performance evaluation of the proposed algorithm is analyzed in Section 5. Section 6 is the conclusion.

2. Related Work

Resource management is a significant issue in cloud computing, as the on-demand resources offering manner. There are various studies on resource management in cloud computing [4–8].

2.1. Game Theoretic Resources Management in Cloud. Recently, game theory has been applied to solve resource allocation problems in cloud computing. Ye and Chen study noncooperative games for the load balancing and virtual machine placement problem [9]. They focus on the existence of Nash equilibrium and care little about the solution for an optimal allocation strategy. Hassan et al. propose a method for the distributed resource allocation problem in federated cloud which studies both noncooperative and cooperative games [10]. They show that the cooperative allocation game has a stronger motivation for providers to contribute resources. But their work models the resources as a single type, while our work considers the allocation problem in multiresource environment.

2.2. Fair Resources Allocation. The coordination of resource sharing is also one underlying challenge for resource allocation in clouds. Many works have studied the fair allocation so far, for instance, fair scheduler for Hadoop, which divides resources as fixed-size partitions, or slots [11]. Another popular fair policy is the max-min fairness, which tries to maximize the minimum resource each user received. Waldspurger enhances this approach by providing a weighted max-min fairness model to support some policies which consider different factors, like priority, reservation, and deadline [12]. Some approaches have recently been proposed to quantify fairness [13]. However, most of them just investigate the fairness of single type resource allocation problem.

The fair allocation problem for multiple types of resources allocation has been studied by Ghodsi et al. [3]. They present a dominant resource fairness approach, which addresses the problem by computing the dominant share of each user. Parkes et al. [14] extend the DRF approach by leveraging a technical framework and study the indivisibilities allocation. They prove that the mechanism satisfies three properties of fairness. Their work still has some drawbacks as they do not take the resource wastage into consideration.

Our work makes use of DRF approach to measure the fairness of resource allocation, as well as exploiting a way to improve the resource utilization for greater optimization.

2.3. Efficient Resource Allocation. For a cloud with a large amount of heterogeneous physical servers in data center, how to achieve efficient resource consumption is another interesting direction of resource allocation [15, 16].

Steinder et al. investigate the resource allocation for a heterogeneous mix of workloads and present a system to manage data center to increase the resource consumption of servers [17]. Di and Wang maximize resource utilization and optimal execution efficiency by proposing a novel scheme DOPS [18]. Cardosa et al. study the efficiency resource allocation of MapReduce clouds [19]. They propose a spatiotemporal tradeoff technique to scale MapReduce clusters dynamically to improve energy consumption while simultaneously improving performance. Jord et al. [20] show a resource-aware multiple job scheduling technique for MapReduce that improves resource utilization while meeting completion time goals. Sandholm and Lai [21] present a resource allocation system that uses user-assigned priorities to offer different service levels and adjusts resource allocations dynamically to fit the requirements.

In contrast to these studies, our work tries to place virtual machines on proper physical servers to minimize resource fragments and achieve the spatial efficiency.

Although some existing researches study on the tradeoff between fairness and efficiency [22], few of them consider the case for multiple types of resources allocation in cloud. Joe-wong et al. develop a framework to address the fairness-efficiency tradeoff with multiple resources [23]. The limitation of this approach is that they should characterize the parameters to ensure the fairness properties satisfied.

The resource allocation mechanism devised in our work exploits incorporation of multiresource nature of virtual machine so as to avoid wastage, while also incentivizing users to share resources in fair way.

3. Resource Management System Modeling

Each cloud provider has a large scaled and distributed data center with heterogeneous physical servers and provides the numerous computational resources as a pay-per-use business models. Infrastructure-as-a-Service providers let users apply for virtual machines and charge them for the occupied time. A VM is created by Xen, VMware, or hypervisor on a physical server. Cloud users deploy their high-performance applications on a cluster of VMs to accomplish their missions (web services or MapReduce jobs), which are called jobs in our work. Cloud providers typically offer a group of possible

VM types to simplify selection for users, and each type is defined by specifying the number of CPU cores, the memory size, the storage size, and the quantities of other resources.

Since the VMs required by different users are heterogeneity and vary in time, providers have to adjust their resource allocation decisions dynamically. To this end, a resource management system for cloud is designed first.

3.1. Resource Management System. We are interested in providing a fair and effective resource allocation mechanism on a distributed and complex cloud system; thus, a resource management system is necessary to centralized control and coordinate the physical resources.

Figure 1 illustrates the architecture of the cloud resource management system proposed in this work. This resource management system has four components which include the register center (RC), cloud environment monitor (CEM), infrastructure management (IM), and control center (CC). The four mentioned components are described as follows.

(i) RC: every physical server in cloud data center should register its information to RC for connection and management.

(ii) CEM: this component retrieves information like host names, IP addresses about physical servers, monitors their statuses (starting, running, shutdown) and the consumption of CPU, memory, and disk storage.

(iii) IM: it is responsible for deploying and managing the virtualized infrastructures, such as creating and releasing virtual machines.

(iv) CC: it is the computing center to provide the most appropriate decision about resource allocating.

CEM is monitoring the statuses and resource consumptions for physical servers registered in RC. Once a new physical server started to join the cloud, the information like MAC address, IP address will be registered to RC. When a user sends a service request to cloud, the requirements of resources in this request will be received by CC. CC makes an intelligent resource allocation decision based on the information collected by CEM. The allocation decision is executed by IM to manage the physical servers and place the virtual machines.

This paper proposes an adaptive resource allocation mechanism in cloud environment, which is the problem of mapping a limited quantity of resources to independent users to finish their jobs. In our resource management system, the allocation of resources is in a time-slotted paradigm. The dynamically arriving user requests of current time slot are recorded and will be served for resource allocation at the start of next time slot. Each start of a time slot is called a decision moment. If a user request cannot be served by the idle resources in current time slot, it will be deferred to the next time-slot or trigger the management of physical servers, which is not considered in our work.

3.2. Mathematical Model. Suppose there are p physical servers available in a cloud cluster and each is denoted as m, where $1 \le m \le p$. Consider k kinds of resources in our work and the available resources of each physical server m can be described as a capacity vector $\overrightarrow{C}^{(m)} = (C_1^{(m)}, C_2^{(m)}, \ldots, C_j^{(m)}, \ldots C_k^{(m)})$, which is monitored by CEM. Each kind of resource is denoted as j. For example, (4,8,40) illustrates that a physical server has 4 CPUs, 8 GB memory, and 40 GB disk storage available.

The job submitted by user i is denoted as J_i, where $i \in \{1, 2, \ldots, s\}$. A variety of VM types are predefined by the cloud provider and a type is encoded by the vector $\overrightarrow{r_i} = (r_{i1}, r_{i2}, \ldots, r_{ij}, \ldots r_{ik})$. Each job applies for a cluster of VMs with the same type to be fully executed. In general, the performance will be improved when the number of VMs assigned to this job increases. However, the cost of cloud provider will also increase to create more VMs. Therefore, the cloud provider should make decisions on the number of VMs assigned to each job.

In Figure 2, we consider two physical servers with capacity vectors (4, 8, 40), (4, 6, 50). Three users apply for three types of VMs which can be described as (2, 4, 20), (1, 1, 10), and (2, 2, 10). Resources like CPU, memory, storage of a physical server will be occupied when a virtual machine created on this server. At each decision moment, resource management system checks the capacity states of all physical servers in data center and analyzes all the user requests to generate the resource requirement matrix.

Definition 1 (resource requirement matrix). Resource requests submitted by different users can be defined as a matrix. Let R be an $s \times k$ dimensional matrix whose rows specify the VM type each user needs, while columns describe the amount of different resources (as shown in Figure 2):

$$R = \begin{pmatrix} \overrightarrow{r_1} \\ \overrightarrow{r_2} \\ \vdots \\ \overrightarrow{r_s} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1k} \\ r_{21} & r_{22} & \cdots & r_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{s1} & r_{s2} & \cdots & r_{sk} \end{pmatrix}. \quad (1)$$

The goal of the resource allocation problem, given the resource requirement matrix and the capacity sets of physical servers, is to determine a reasonable mapping from resources to cloud users. In other words, different kinds of resources each physical server has should be fair and effectively distributed to all users to create their required VMs.

Definition 2 (allocation decision). For the physical server m, a possible resource allocation state can be described as an allocation matrix $A^{(m)}$:

$$A^{(m)} = \begin{pmatrix} \overrightarrow{a}_1^{(m)} \\ \overrightarrow{a}_2^{(m)} \\ \vdots \\ \overrightarrow{a}_s^{(m)} \end{pmatrix} = \begin{pmatrix} a_{11}^{(m)} & a_{12}^{(m)} & \cdots & a_{1k}^{(m)} \\ a_{21}^{(m)} & a_{22}^{(m)} & \cdots & a_{2k}^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{s1}^{(m)} & a_{s2}^{(m)} & \cdots & a_{sk}^{(m)} \end{pmatrix}, \quad (2)$$

where $a_{ij}^{(m)}$ denotes the amount of resource j on physical server m allocated to user i.

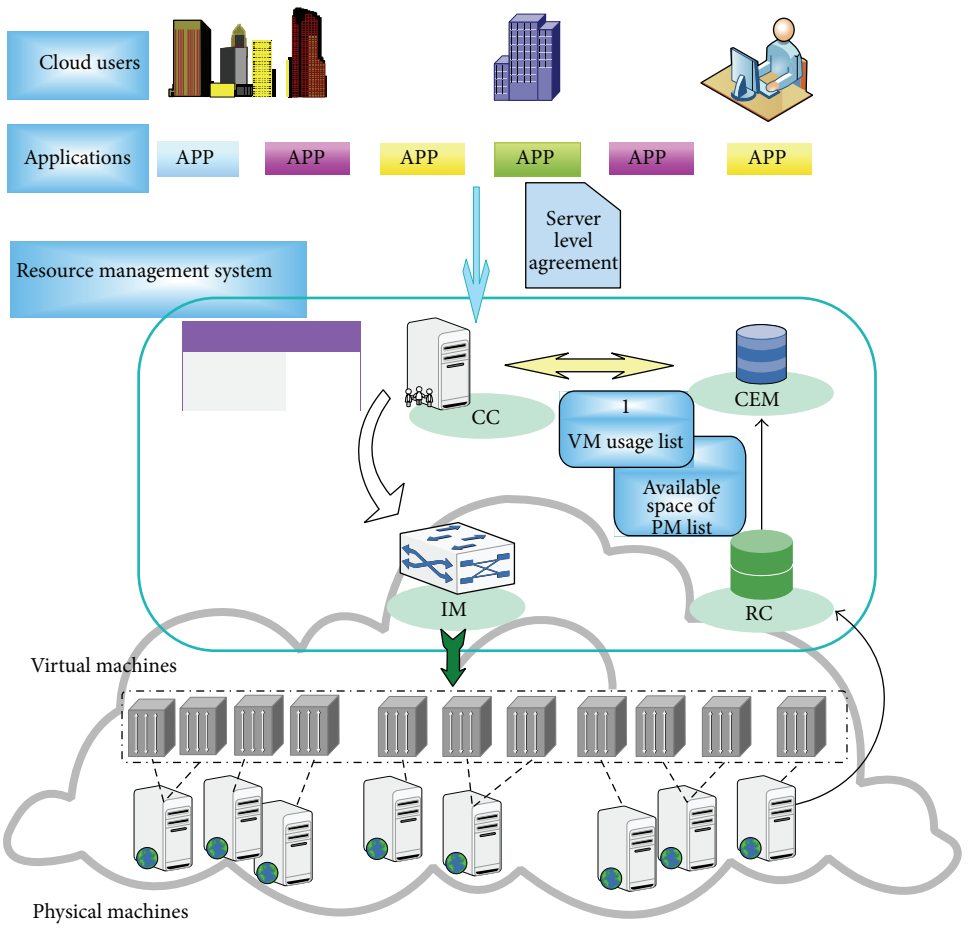


Figure 1: A framework of cloud resource management system.

An allocation decision A is a collection of a possible allocation status of every physical server based on the resource requirement matrix:

$$A = \left\{A^{(1)}, A^{(2)}, \ldots, A^{(m)}, \ldots A^{(p)}\right\}. \quad (3)$$

Figure 2 gives an example of allocation decision. If the Physical Server 1 creates one type-$\overrightarrow{r_1}$ VM and one type-$\overrightarrow{r_2}$ VM, then the allocation matrix is shown as $A^{(1)}$. Each of the two physical servers in Figure 2 has its own allocation matrix and consists of an allocation decision $A = \{A^{(1)}, A^{(2)}\}$.

Furthermore, the total number of resource j allocated to user i is denoted by $\varphi_{ij} = \sum_m a_{ij}^{(m)}$. The parameters and their description are summarized in Notation section.

4. Game Theoretic Resource Allocation

Each user in a cloud asks for a type of VM to run its job. The execution of a job involves multidimensional resources, and the resource requirements differ from job to job. For example, a data mining job needs high capacity of disk to store a large number of data while a calculating job might need more CPU than disk to get a result.

In order to support elastic multiresource consumption, we propose a fairness-utilization tradeoff game algorithm (FUGA), which makes an optimal tradeoff between fairness and efficiency.

4.1. Fair Allocation. In this paper, the fair allocation problem is considered for multiples types of resources. For a single type of resources, fair allocation means each user has equal share of resources. However, in multiresource environment, since users have heterogeneous requirements for different types of resources, resources should be assigned to users in proportion to their requirements. Each user has a maximum share fraction of total capacity among different resources which is called dominant share. The major goal of fair allocation considered in our work is to equalize the dominant share of each user.

Three widely used properties should be satisfied to achieve fair allocation [3].

Definition 3 (sharing incentive). Sharing incentive means the amount of resource each user should receive is at least as much as simply splitting the total resources equally.

Definition 4 (envy-freeness). Envy-freeness is the property that no user prefers to the allocation of another user.

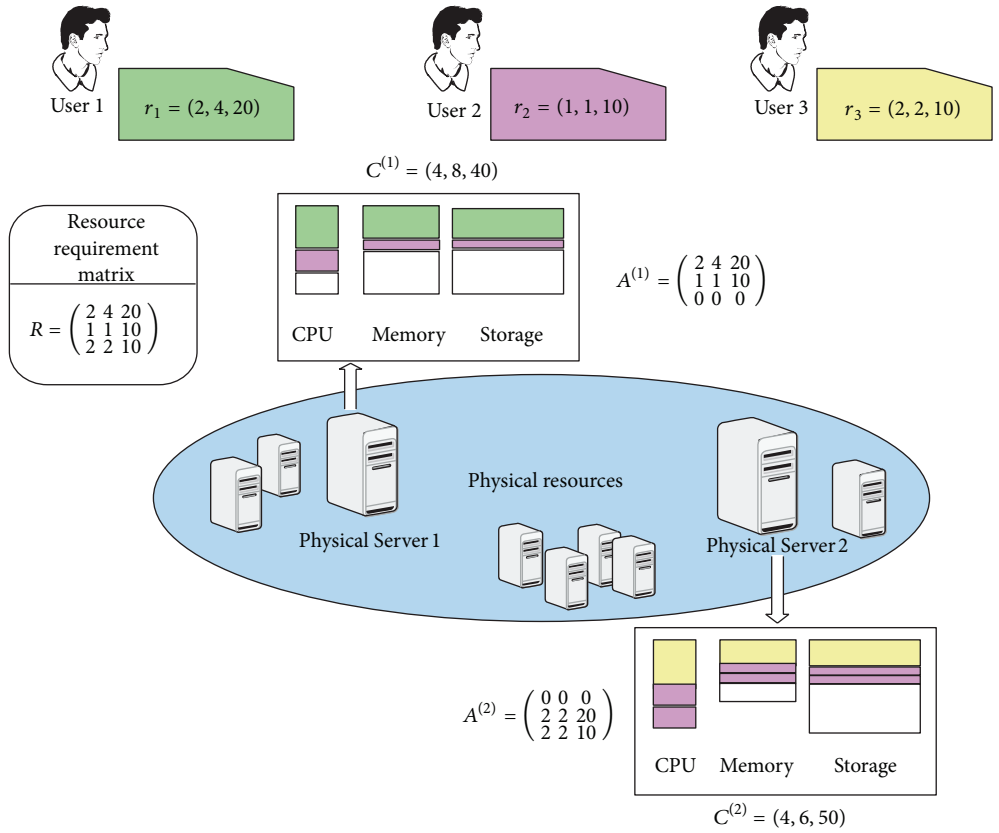


Figure 2: An example of cloud resource allocation.

Definition 5 (Pareto efficient). It should be impossible to increase the resource amount of a user without decreasing the allocation of another user.

The fairness of multiple resources sharing is measured by extending the dominant resource fairness (DRF) mechanism that Ghodsi et al. put forward at 2011. In words, to mathematically gauge the fairness of a resource allocation mechanism, the DRF is set to be the benchmark of fair allocation. Each allocation decision may have a deviation contrast to the fair allocation, called fairness variance.

Given a resource requirement matrix R and the summation of the total resources for all physical servers $C = (\sum_m C_1^{(m)}, \sum_m C_2^{(m)}, \ldots, \sum_m C_j^{(m)}, \ldots, \sum_m C_k^{(m)})$, the first step is to normalize the requirement matrix. The normalized matrix is denoted as Ψ:

$$\Psi = \begin{pmatrix} \psi_{11} & \psi_{12} & \cdots & \psi_{1k} \\ \psi_{21} & \psi_{22} & \cdots & \psi_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{s1} & \psi_{s2} & \cdots & \psi_{sk} \end{pmatrix} = \begin{pmatrix} \frac{r_{11}}{C_1} & \frac{r_{12}}{C_2} & \cdots & \frac{r_{1k}}{C_k} \\ \frac{r_{21}}{C_1} & \frac{r_{22}}{C_2} & \cdots & \frac{r_{2k}}{C_k} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{r_{s1}}{C_1} & \frac{r_{s2}}{C_2} & \cdots & \frac{r_{sk}}{C_k} \end{pmatrix}. \quad (4)$$

Secondly, as mentioned before, the dominant share of a user is the largest fraction of any kinds of resources allocated to that user. Let $d_{ij} = \psi_{ij}/(\max_j \psi_{ij})$ be the normalized demands, and $\lambda = 1/(\max_j \sum_i d_{ij})$ is the dominant share.

Consider the example in Figure 2, the total amount of available resources on the two physical servers is $C = (C_1^{(1)} + C_1^{(2)}, C_2^{(1)} + C_2^{(2)}, C_3^{(1)} + C_3^{(2)}) = (8, 14, 90)$. The resource requirement matrix R is $\begin{pmatrix} 2 & 4 & 20 \\ 1 & 1 & 10 \\ 2 & 2 & 10 \end{pmatrix}$. The normalized matrix is deduced as $\Psi = \begin{pmatrix} 1/4 & 2/7 & 2/9 \\ 1/8 & 1/14 & 1/9 \\ 1/4 & 1/7 & 1/9 \end{pmatrix}$, and then the normalized demands are $d_{11} = 7/8, d_{12} = 1, d_{13} = 7/9, d_{21} = 1, d_{22} = 4/7, d_{23} = 8/9, d_{31} = 1$, and $d_{32} = 4/7, d_{33} = 4/9$. The dominant share is $\lambda = 1/(7/8 + 1 + 1) = 8/23$.

Fairness variance is defined to measure the fairness of a resource allocation. Let x_{ij} denote the amount of resource j allocated to user i in a real allocation. Fairness variance can be calculated mathematically as follows, and $\alpha \in R$ is a parameter:

$$v(A) = \left(\sum_i \sum_j \left(\left| \frac{x_{ij}}{C_j} - \lambda \cdot d_{ij} \right| \right)^{\alpha - 1} \right)^{1/\alpha}. \quad (5)$$

4.2. Resource Utilization. We next turn our attention to the resource utilization problem. During the running time, the resources of physical severs may not be fully used. Consider the example in Figure 2, once the Physical Server 1 creates

one type-$\overrightarrow{r_1}$ VM and two type-$\overrightarrow{r_2}$ VMs, 2 G memory will be left over as a resource fragment. Given the set of VMs along with their resource configurations, how to find the most efficient way to pack these VMs on the physical servers so as to minimize the spatial wastage of resources is considered here.

In multiresource environment, to improve the resource utilization, resource consumption on each resource dimension should be concerned. To address this challenge, our approach improves the resource utilization rate of physical servers based on two considerations. Firstly, the max-min approach, which means we should maximize the minimum consumption among the multiple resources of each physical server, is applied here. Secondly, the utilization of a physical server can be optimized by minimizing the uneven consumption in the face of multidimensional resources, since most of the resource fragments are caused by the unequal multiresource requirements [24].

As said earlier, it is critical to consider the bottleneck resources consumption among the multiple types of resources. Let us denote the vector $\overrightarrow{\Pi}_i^{(m)} = (\pi_1^{(m)}, \pi_2^{(m)}, \ldots, \pi_j^{(m)}, \ldots, \pi_k^{(m)})$ to represent the initial resource space of a physical server when no VM is created on it. Then the minimum resource utilization function of physical server m can be expressed as

$$u_{\min}^{(m)} = \min_j \left\{u_j^{(m)}\right\} = \min_j \left\{1 - \frac{C_j^{(m)} - \sum_i x_{ij}^{(m)}}{\pi_j^{(m)}}\right\}. \quad (6)$$

$x_{ij}^{(m)}$ is the amount of resource j allocated to user i on physical server m.

More formally, skewness is introduced to quantify the unevenness for the utilization of different resources. The reduction of skewness can be positive to combine multiple types of resources better and improve the utilization:

$$\text{ske}(m) = \sqrt{\sum_{j=1}^{k}\left(\frac{u_j^{(m)}}{\overline{u^{(m)}}} - 1\right)^2}, \quad (7)$$

where u_j^m is the utilization of resource j and $\overline{u^m}$ is the average utilization of all resources for physical server m.

In order to achieve a high utilization of computing resources, the cloud provider tries to coplace VMs on available machines such that the resource requirements on one server are complementary to each other. This VM placement problem can be reduced to a multidimensional bin packing problem. Several heuristic algorithms such as Best-Fit, First-Fit, or Random-Fit are typically used to address it. In our work, a precombination approach introduced in Section 4.3 is proposed to optimal the virtual placement problem.

4.3. The Fairness-Utilization Tradeoff Game Algorithm. In this section, a game theory approach to resource allocation is presented, aiming at keeping a fair allocation as well as reducing the amount of resource fragments. The game model for resource allocation problem is described first, followed by the proposal of the FUGA algorithm.

(1) Resource Allocation Game. Game theory is a mathematical study of strategy which attempts to determine the interactions among all game players to ensure the best outcomes for themselves [25–28]. A game consists of three factors, that is, a set of players, all the possible strategies each player will choose, and the specified utilities of players associated with the strategy performed by every player. At each step, players choose one of their strategies and get a utility in return. Each player of a game tries to maximize its own utility by choosing the most profitable strategy against other players' choices. Nash equilibrium is a central notion of game theory which means in this situation no player can get more utilities by changing its strategy.

A specification of a game is an extensive game which provides the sequencing of all players' possible strategies and their decision points. A finite extensive game with perfect information has a finite set of players, and each player knows the information of other players' strategies and all possible utilities. A subgame perfect Nash equilibrium (SPNE) is a solution such that players' strategies constitute a Nash equilibrium in every subgame of an original game.

In our work, the resource allocation problem is modeled as a finite extensive game with perfect information. Physical servers with idle resources are modeled as the selfish players and each player has a limited number of possible allocation matrices.

The following symbols are introduced to define the resource allocation game.

Definition 6. A resource allocation game is represented as a four-tuple vector $G = (P, R, A, U)$.

(i) P is the players in the allocation game.

(ii) R refers to the resource requirement matrix of users.

(iii) A are the sets of players' strategies.

(iv) U is the utility function of game players.

At decision moment, CC gets the resources consumption information of each physical server in data center from CEM. P is represented for the set of physical servers with idle resources and each server is associated with a capacity vector. All the users' requests for cloud resources CC receives during last time-slot are analyzed and transformed to the resource requirement matrix R.

For a physical server, there are a variety of possible combinations to be fulfilled by different types of VMs without exceeding the capacity. A combination of physical server m can be denoted as a $\text{com}_x(m) = \langle c_{x1}, c_{x2}, \ldots, c_{xs}\rangle$. For instance, the cloud users ask for three VM types $\overrightarrow{r_1}$, $\overrightarrow{r_2}$, and $\overrightarrow{r_3}$ corresponding to vectors (2, 4, 20), (1, 1, 10), and (2, 2, 10) in Figure 2, and physical sever 1 has (4, 8, 40) capacity of spare resources. $\langle 1, 1, 0\rangle$ means one VM of type $\overrightarrow{r_1}$ and one VM of type $\overrightarrow{r_2}$ can be created on physical server m.

In this resource allocation game, the physical servers with idle resources are game players, and they are individual

rationality to maximize their own utilities. Based on the discussions in previous, the design of the utility function has a crucial impact on players' choices and the result of the game. In our allocation model, one global objective of this allocation game is to share resources impartiality. Furthermore, based on the efficient principle each individual player tries to minimize their resource wastage, that is, they prefer to choose those combinations with high utilization. To exploit fair resource sharing and also take the maximization of resource utilization rate into account, a fairness-utilization tradeoff utility function is designed as follows:

$$U^{(m)}(A) = \operatorname{sgn}(1-\alpha)\cdot v(A) - \operatorname{ske}(m). \tag{8}$$

α is a coefficient to affect the weights of fairness and utilization. $v(A)$ is the fairness variance (defined in formula (5)) and $\operatorname{ske}(m)$ is the skewness which reflects the unevenness for the utilization of different resources (defined in formula (7)). The less fairness variance an allocation decision A gets, the more utilities players gain. Similarly, each physical server prefers to choose the combination with less skewness to optimal its own utility.

Each player of this game aims to choose a strategy to maximize its own utility so that the goal of a resource allocation game would be naturally considered as the following optimization problem:

$$\begin{aligned} &\text{Maximize} \quad U^{(m)}(A) \\ &\text{Subject to} \quad \sum_i \sum_m a_{ij}^{(m)} \le C_j \\ &\qquad\qquad a_{ij}^{(m)} \ge 0. \end{aligned} \tag{9}$$

$A^* = \{A^{(1)^*}, A^{(2)^*}, \ldots, A^{(m)^*}, \ldots A^{(p)^*}\}$ is the Nash equilibrium of a resource allocation game which means for all $m, U(A^1, A^2, \ldots, A^{m^*}, \ldots, A^p) > U(A^1, A^2, \ldots, A^{m'}, \ldots, A^p)$.

(2) The FUGA Algorithm and Its Properties. The FUGA algorithm is presented to result in an optimal allocation decision for this resource allocation game.

Step 1. Firstly, to achieve a high level resource utilization rate by reducing the resource fragments generated in the virtual machine placement process, a precombination approach is proposed to provide a set of possible strategies for each game player. A precombination phase is defined in this approach to compute any possible coordinate placement combinations for each physical server. For example, to place these three types of VMs together on the Physical Server 1 in Figure 2 without exceeding the capacity, all the possible combinations are $\langle 2,0,0\rangle$, $\langle 1,2,0\rangle$, $\langle 1,1,0\rangle$, $\langle 1,0,1\rangle$, $\langle 0,2,1\rangle$, $\langle 0,1,1\rangle$, $\langle 0,4,0\rangle$, $\langle 0,3,0\rangle$, $\langle 0,2,0\rangle$, $\langle 0,1,0\rangle$, $\langle 0,0,2\rangle$, and $\langle 0,0,1\rangle$.

Step 2. To facilitate efficient selection of resource allocation, the minimum resource utilization of each combination is calculated by making use of formula (6). Some of the combinations may result in efficient resource consumption across all physical resources, while many others will leave much wasted space. For each physical server, we rank the combinations by their minimum resource utilization (the maximum resource consumption threshold of a physical node is also predefined to reject those overload combinations). The minimum resource utilization of a combination is

$$u_{\min}^{(m)}\left(\operatorname{com}_x(m)\right) = \min_j \left\{ 1 - \frac{C_j^{(m)} - \sum_i \left(c_{xi}\cdot r_{ij}\right)}{\pi_j^{(m)}} \right\}. \tag{10}$$

We describe the ranked combinations mathematically as an order $O^{(m)}$ for physical server m. For $x, y \in N$, for all $\operatorname{com}_x(m), \operatorname{com}_y(m) \in O^{(m)}$, if $x < y$, $u_{\min}^{(m)}(\operatorname{com}_x(m)) \ge u_{\min}^{(m)}(\operatorname{com}_y(m))$. For the two combinations with equal value of minimum resource utilization, the order between them can be ranked randomly.

Since each physical server with sufficient idle space has a set of possible combinations calculated in advance and ordered by resource utilization, only the top of η combinations is chosen as the game strategy set of this player. Considering the performance of computing, the value of η will decrease along of the number of game players increasing. Then the utilization of ηth combination in order $O^{(m)}$ is denoted as

$$\min\left(O^{(m)}\right) = \min_{\text{com}}\left\{u_{\min}^{(m)}(\text{com}(m))\right\} = u_{\min}^{(m)}\left(\text{com}_\eta(m)\right). \tag{11}$$

If η is set to be 3 for the example of Figure 2 and the initial resources space of the Physical Server 1 is (6, 16, 100), then the $O^{(1)}$ is $\{\langle 2,0,0\rangle, \langle 1,2,0\rangle, \langle 1,0,1\rangle\}$. Similarly, the $O^{(2)}$ is $\{\langle 1,2,0\rangle, \langle 1,0,1\rangle, \langle 0,0,2\rangle\}$, where the initial resources space of the Physical Server 2 is (8, 20, 120). Moreover, the minimum resource utilizations of the two servers are $\min(O^{(1)}) = 0.875$, $\min(O^{(2)}) = 0.75$.

Step 3. Each game player has a set of possible combinations to choose now. Once all players picked up one of their combinations, the allocation matrix $A^{(m)}$ could be rewritten as

$$A^{(m)} = \operatorname{com}(m)^T \cdot R = \left(c_i^{(m)}\cdot r_{ij}\right)_{s\times k}. \tag{12}$$

For an allocation A, we add up the amount of resources for each user j and denote the total as

$$\varphi = \sum_m A^{(m)} = \left(\left(\sum_m c_i^{(m)}\right)\cdot r_{ij}\right)_{s\times k}. \tag{13}$$

Therefore, to determine the optimal resource allocation decision, the utility function that derives from the fairness

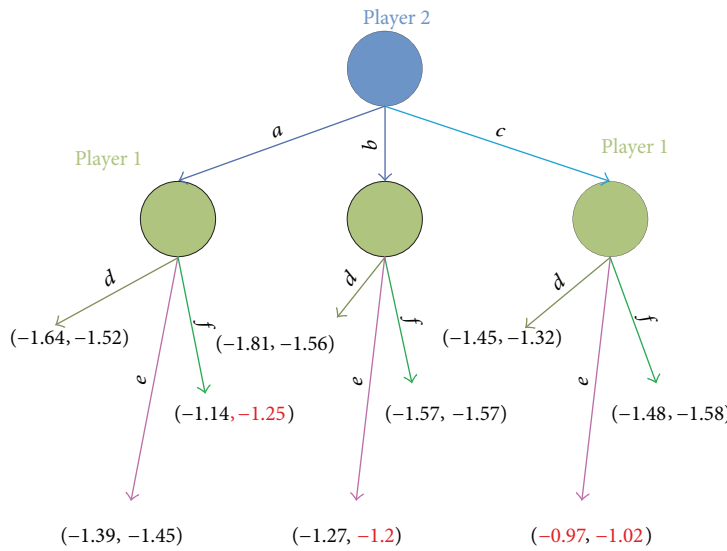


Figure 3: The extension-form game tree.

variance function and skewness in formula (8) can be transformed to

$$U^{(m)}(A) = \operatorname{sgn}(1-\alpha)\cdot v(A) - \operatorname{ske}(m) = \operatorname{sgn}(1-\alpha)\left(\sum_i\sum_j\left(\left|\frac{\varphi_{ij}}{C_j} - \lambda\cdot d_{ij}\right|\right)^{\alpha-1}\right)^{1/\alpha} - \sqrt{\sum_{j=1}^{k}\left(\frac{u_j^{(m)}}{\overline{u^{(m)}}} - 1\right)^2} \quad (\alpha > 1). \tag{14}$$

Supposing the Physical Server 1 and Physical Server 2 both choose their second combination, then

$$\varphi = \left((1,2,0)^T + (1,0,1)^T\right)\cdot R = \begin{pmatrix} 4 & 8 & 40 \\ 2 & 2 & 20 \\ 2 & 2 & 10 \end{pmatrix}. \tag{15}$$

Assuming that $\alpha = 2$, we calculate the utility of each physical server and get results $U^{(1)} = -1.2$, $U^{(2)} = -1.27$.

The resource allocation game is modeled as the interaction of physical servers to make choices with perfect information. Each extensive game can be represented as an extension-form game tree. Physical servers take actions in the ascending order of $\min(O^{(m)})$ and a game tree for this extensive game is completed based on the utility function. In the case we considered in Figure 2, the Physical Server 1 and Physical Server 2 are two game players. Physical server 1 has three strategies $\langle 2,0,0\rangle$, $\langle 1,2,0\rangle$, and $\langle 1,0,1\rangle$ represented as d, e, and f, while the Physical Server 2 has three strategies $\langle 1,2,0\rangle$, $\langle 1,0,1\rangle$, and $\langle 0,0,2\rangle$ represented as a, b, and c. The Physical Server 2 takes action first, followed by the Physical Server 1, based on the ascending order of $\min(O^{(m)})$. The value pairs indicate the utilities of the Physical Server 2 and Physical Server 1 corresponding to the strategy each one choose. Figure 3 shows the extension-form game tree.

Step 4. Backward induction is a quite straightforward solution to find an SPNE for these extensive-form games with perfect information [29]. It is a recursive mathematical construction for players to take rational behavior in a finite extensive game with perfect information [30, 31].

In Figure 3, let us start from the bottom of the extensive-form game tree. Consider the subgame of player 1; suppose player 2 chooses the strategy a; the strategy f is better than others for player 1 to get a higher utility. Analogously, player 1 will choose strategy e when player 2 chooses the strategy b or strategy c. Player 1 knows this, so it will choose strategy c to maximize its own utility. T is procedure which produces a SPNE is called backward induction. Thus, we get the SPNE strategy (c, e) which means player 2 will choose its third combination and player 1 picks up its second one.

The pseudocode implementation of FUGA is given in Algorithm 1. The backward induction approach is used to carry out the optimal placement of the VMs considering a tradeoff between the fairness and utilization. This algorithm starts from last two servers, getting the Nash equilibrium for the subgame. Following the descending order of this game tree, in each turn we take a new physical server into considering and recalculate the utilities of the new subgame. According to this, for each considered physical server, the most suitable combination of VMs with the highest utility will be placed on it.

Theorem 7. *FUGA has at least one pure strategy Nash equilibrium.*

Proof. According to the Zermelo's theorem, a finite game with perfect information has a pure strategy Nash equilibrium, and for a finite extensive game with perfect information, there

Input: $\left\{\overline{C}^{(m)}\right\}, R$
Output: A^*
(1) Initialization: *combinLists*, *selectedServerList*, *selection* $[p+1]$
(2) ***Step 1.*: Pre-combination Phase**
(3) // Each physical server with idle resources is a game player
(4) $P \leftarrow \{1, \ldots, m, \ldots, p\}$
(5) **for each** physical server m **do**
(6) List any possible coordinate placement combinations of this server to be fulfilled by different types of VMs without exceeding the capacity in the $combinList_m$
(7) *combinLists*.add($combinList_m$)
(8) **end for**
(9) ***Step 2.*: Strategies Set for Each Player**
(10) **for each** physical server m **do**
(11) Pick up top η of combinations $O^{(m)} = \{\text{com}_1, \text{com}_2, \ldots, \text{com}_\eta\}$ and calculate $\min(O^{(m)})$
(12) // each $com_x(m)$ can be represented as an allocation matrix $A_x^{(m)}$
(13) **end for**
(14) ***Step 3.*: Generate the Extension-form Game Tree**
(15) The orginal array $\left[\min\left(O^{(1)}\right), \ldots, \min\left(O^{(p)}\right)\right]$ is rearranged in a non-decreasing order with indices $\left[i_1, \ldots, i_p\right]$ such that $\min(O^{(i_1)}) \leq \cdots \leq \min(O^{(i_p)})$
(16) The game players take action as the order of $\left[i_1, \ldots, i_p\right]$
(17) ***Step 4.*: Find the SPNE for a game *G***
(18) **for each** strategy $A_x^{(i_{p-1})}, A_y^{(i_p)}$ of physical server i_{p-1}, i_p **do**
(19) Calculate the utility pair $\left(U^{(i_{p-1})}[x][y], U^{(i_p)}[x][y]\right)$
(20) **end for**
(21) $\max[x] \leftarrow \arg\max_y U^{(i_p)}[x][y]$
(22) $selection\left[i_{p-1}\right] \leftarrow \arg \text{Max}_x U^{(i_{p-1})}[x][\max[x]]$
(23) $selection\left[i_{p-1}\right] \leftarrow \arg \text{Max}_x (\max[x])$
(24) Add i_{p-1}, i_p to the *selectedServerList*
(25) **for each** physical server m from i_{p-1} to 1 **do**
(26) Add up the total amount of resources φ for physical servers in *selectedServerList*
(27) **for each** strategy $A_x^{(m)}$ of physical server m do
(28) Calculate the ske(m) if $A_x^{(m)}$ is chosen
(29) Add up the total allocated resource $\varphi = A^m + \varphi$
(30) Calculate the $v(A)$
(31) utilityCalculation(ske(m), $v(A)$)
(32) **end for**
(33) $selection[m] \leftarrow \arg \text{Max}_x \left(U^{(m)}[x]\right)$
(34) Add server m to *selectedServerList*
(35) **end for**
(36) The best strategy of each player m in $selection[m]$ can be represented as an allocation matrix $A^{(m)}$, and $A^* = \left\{A^{(1)}, \ldots, A^{(m)}, \ldots, A^{(p)}\right\}$

Algorithm 1: The FUGA algorithm.

always exists an SPNE. If no player has the same utilities at any two terminal nodes of subgames, then a unique SPNE can be derived from backward induction. □

Now we prove that FUGA satisfies the three required properties for fair allocation described in the previous section.

Theorem 8. *FUGA is sharing incentive.*

Proof. An allocation algorithm satisfies sharing incentives if $U_i(A^{(m)^*}) \geq U_i(\langle C_1^{(m)}/s, C_2^{(m)}/s, \ldots, C_k^{(m)}/s\rangle)$ for all i. Taking the game theory perspective, a Nash equilibrium allocation A^* maximizes the utility of each game player m. In other words, the utility of Nash equilibrium strategy is better than any other strategies; that is, $U^{(m)}(A^{(m)^*}) \geq U^{(m)}(A^{(m)})$. For each independent user i, once the optimal strategy of each player is found out, the utility function $U_i(A^*)$ can be derived as the sum of utilities for those physical servers which provide resources for user i.

Thus, $U_i(A^*) \geq U_i(A)$, $\langle C_1^{(m)}/s, C_2^{(m)}/s, \ldots, C_k^{(m)}/s\rangle \in A$ so that $U_i(A^*) \geq U_i(\langle C_1^{(m)}/s, C_2^{(m)}/s, \ldots, C_k^{(m)}/s\rangle)$. □

Theorem 9. *FUGA satisfies envy-freeness property.*

Proof. An allocation algorithm is envy free if $U_i(A_i^*) \geq U_i(A_{i'})$ for all i, where i and i' are different users. For a strategy profile, its Nash equilibrium means that no player wants to deviate to another strategy. As proven above, $U_i(A^*) \geq U_i(A)$. Therefore, the envy-freeness property is satisfied. □

Theorem 10. *FUGA is Pareto efficient.*

Proof. Assume user i can increase its resource share without decreasing profits of anyone else. When $\alpha > 1$, higher utility represents that the optimal allocation is much closer to the resource demands of user. If we add some resource for user i on physical server m, it is clear that the new allocation has a larger fairness variance. Obviously, the utility of game player m decreased, such that the utility of other user who share the resources on physical server m will also decrease. Thus, *FUGA* satisfies the Pareto efficient property. □

5. Performance Evaluation and Comparison

This section presents a comprehensive evaluation of the resource allocation algorithm proposed in the previous section. The evaluation of fairness is done through a prototype implementation of our FUGA algorithm running on an 8-node cluster first. And then the conduct of Google Trace-driven simulations shows that FUGA is efficient in improving the resource utilization by contrast with the First-Fit Algorithm and the management mechanism of Google cluster.

5.1. Experimental Environment. The experiments to evaluate the performance of fair allocation were done on a small scale cluster with 8 physical nodes which consist of a Dell PowerEdge R910 with two CPUs (Xeon E7-4820 2 GHz 8cores), 32 GB memory, and 300 GB disk storage, three Dell Optiplex9010 with one CPU (i7-3770 3.40 GHz 4cores), 8 GB memory, and 500 GB disk storage, and four Dell Optiplex745 with two CPUs (6600 2.4 GHz 2cores), 4 GB memory, and 200 GB disk storage. Three kinds of resource considered in this experiment include CPU, memory, and disk storage. The simulations were run on a Dell Optiplex9010 with JDK 1.7. To reduce the complexity of simulations, the following assumptions are made: (1) two kinds of resources (i.e., CPU and memory) are considered in our simulations. (2) Each job request submitted by a user indicates the predicted maximum consumption of different resources and will be handled by a cluster of VMs with the same type. (3) The total amount of resources provided for each time slot is previously estimated by cloud provider.

Table 1 shows the five types of VMs with the configurations of different resources considered in our evaluation and the operating systems are Ubuntu-12.04.

5.2. FUGA versus Hadoop Fair Scheduler. This group of experiments aims to show how FUGA dynamically shares resources more close to users' requirements in contrast to the Hadoop fair scheduler.

Hadoop is one of the most popular frameworks for storage and large scale data processing. Hadoop fair scheduler groups jobs into different pools and each pool chooses its jobs based on FIFO or fair sharing [11]. The fair sharing algorithm of Hadoop fair scheduler can be described as follows. When a job arrives, it will be assigned to the pool which has the fewest running jobs to keep an equal number of jobs for all pools. Hadoop framework partitions the resources of physical servers into a number of computation units named slot. The slots which consist of fixed amount of different resources can be treated as single resource type. The assignment of slots is also based on the fair sharing principle so that to get an equal share of resources among jobs.

Table 1: The VM types.

VM type	Resource		
	CPU core	Memory	Disk
Tiny	1	1024 MB	5 GB
Small	1	3072 MB	15 GB
Medium	2	6144 MB	30 GB
Large	4	12288 MB	60 GB
X large	8	24576 MB	60 GB

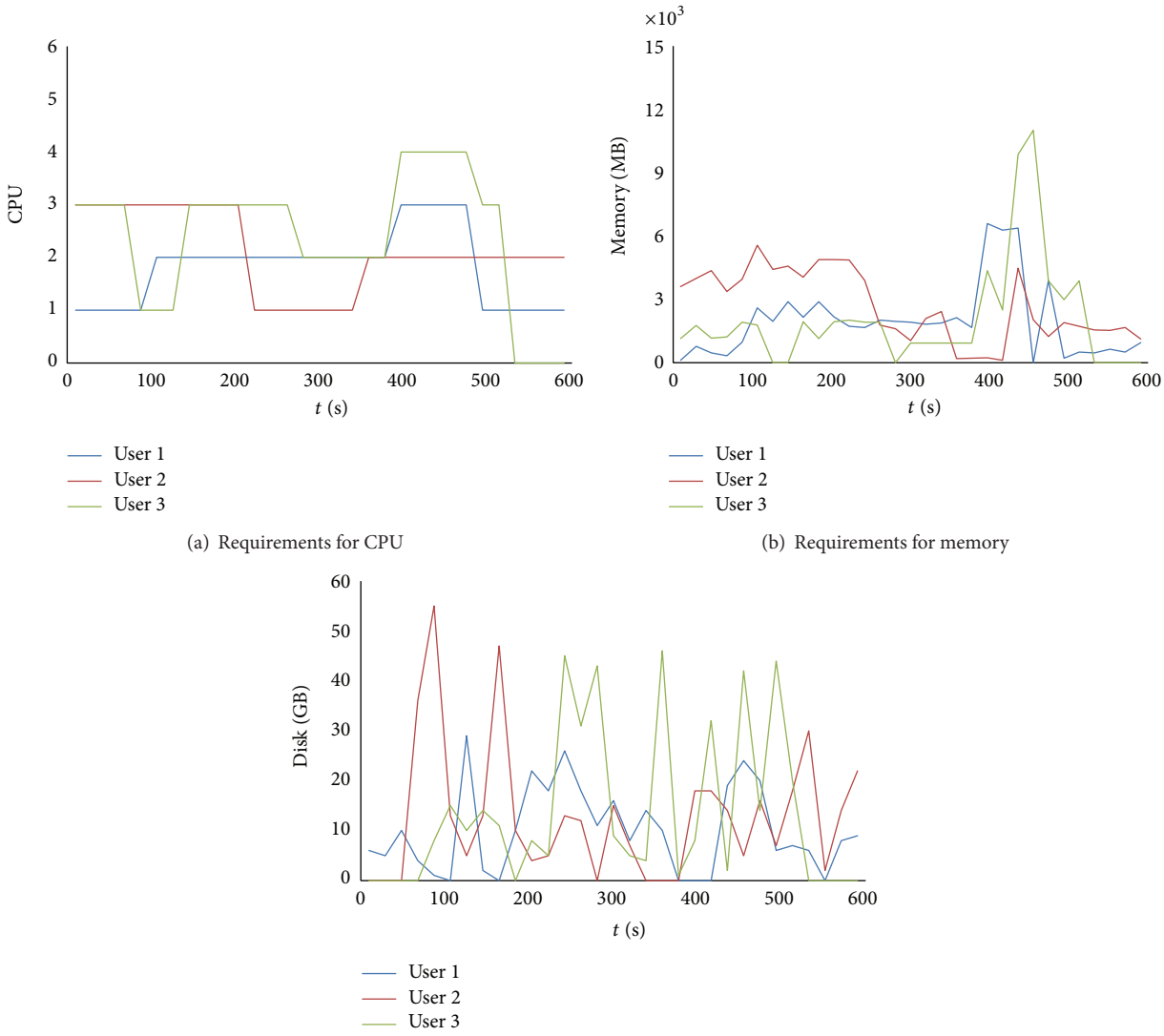
Three services belonging to three users were deployed on a cluster of VMs created on this 8-node cluster. Each service can be divided into a series of MapReduce jobs and has different resource requirements in different phases. The 8-node cluster is initially empty with full capacities. FUGA analyzes the requirements in time and provides optimal resource allocation decisions to create a VM cluster at each decision moment. As a comparison, these three services were also deployed on the Hadoop cluster running on the initialized 8-node. Figures 4(a), 4(b), and 4(c) show the CPU, memory, and disk requirements of each service during a ten-minute interval. The number of combinations each player choose is set to 5 and $\alpha = 2$. Each experiment ran for ten times.

Figure 5 shows the dominant share for Hadoop scheduler and FUGA. As Figure 5(a) shows, FUGA average performs only 13% deviation to equalize the dominant share of three users and can achieve as much as 41% better than what Hadoop scheduler showed in Figure 5(b). It also shows that FUGA satisfies the properties of sharing incentive, envy-freeness, and Pareto efficient.

In contrast to Hadoop fair scheduler, FUGA is aware of the heterogeneity requirements for multiresource environment and significantly more approach to the demands of users on each resource dimension.

5.3. FUGA versus First Fit Algorithm. This section highlights the performance of FUGA on improving the resource utilization rate by analyzing the proportions of allocated resources. The higher proportion an allocation achieves, the less resource are waste. Simulations were conducted using the Google workload trace as the input. This trace collects the data (job workloads, server capacities, execution time, resource utilization, etc.) of Google cluster from about 12,500 machines over the 29 day period.

As α increases, the percent of fairness from FUGA drops. Thus, we set $\alpha = 5$ to make the impact of fairness variance



(a) Requirements for CPU

(b) Requirements for memory

(c) Requirements for disk storage

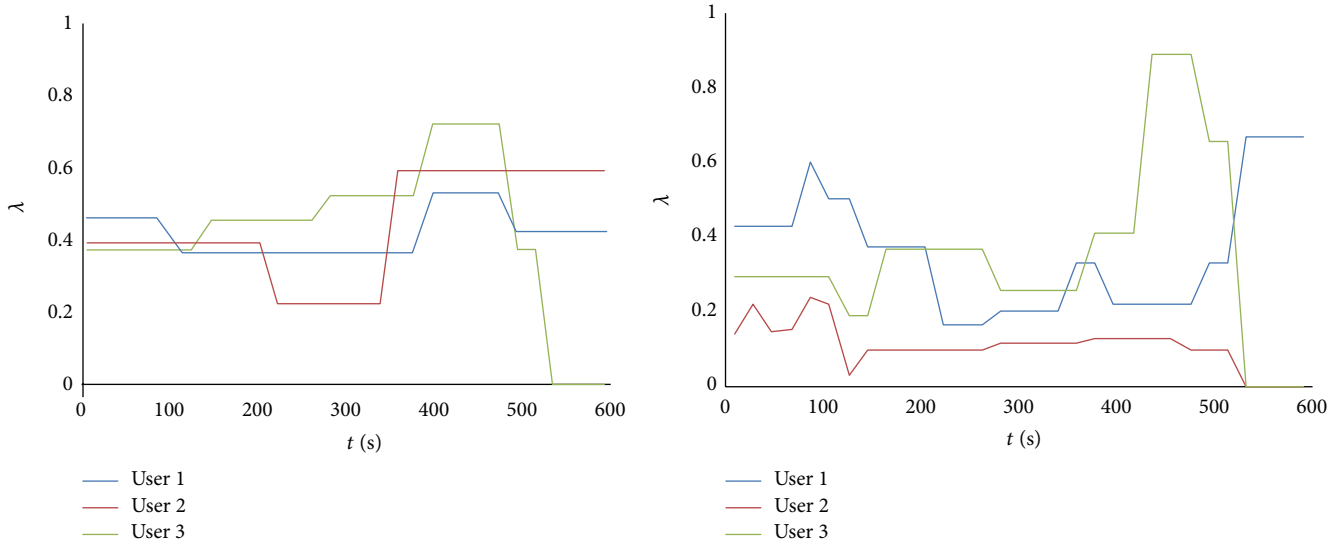
FIGURE 4: Resource consumption for the three services.

and care more about resource efficiency less. The first fit algorithm widely used for VM placement problem in cloud is implemented as a benchmark in our simulation framework. The scale of heterogeneous physical servers is varied from 50 to 700 to observe the proportions of allocated resources for the first fit algorithm [32, 33] and FUGA. By observing Figure 6, for the first fit algorithm, the proportions of CPU, and memory change unregularly as the increasing trend of physical servers. However, FUGA keeps the proportion of allocated CPU fixed to a value of approximately 0.7 where the amount of physical servers reaches 450 and can even allocate 10% more CPU than what the first fit algorithm showed in Figure 6(a). Analogously, the proportion of allocated memory is fixed to about 0.6 where the amount of physical servers reaches 500 shown in Figure 6(b).

As to the large scale environment, it is clear that the less resource fragments produced during the allocation, the higher resource utilization rate we get. Since FUGA can achieve a better performance for proportions of allocated resources if the physical servers scale up to a large number so that to bring a higher utilization rate than the first fit algorithm.

Overall, the results in Figure 6 show that in large scale environment FUGA successfully achieve more efficient resource allocation in contrast to the first fit algorithms. FUGA enables efficient resource utilization of physical resources by considering the differences of requirements among individual resource dimensions and reducing the wastage of resource.

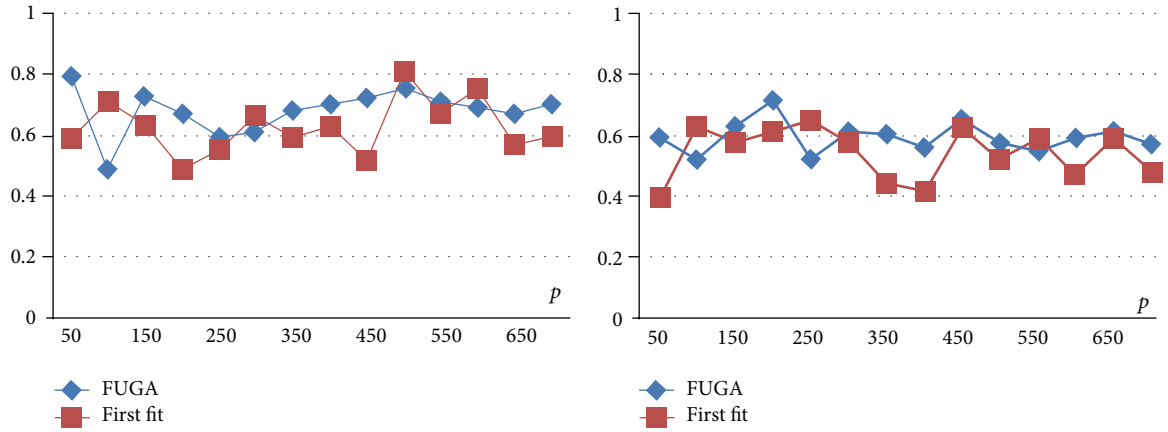
5.4. FUGA versus Google Cluster. To study the performance of efficient allocation by evaluating the resource utilization rate, the total number of physical servers is fixed to 300 in this group of Google trace-driven simulations. The parameter η is set to be 10 and α is 2.



(a) The dominant share for Hadoop scheduler

(b) The dominant share for FUGA

Figure 5: Dominant share for Hadoop scheduler and FUGA.



(a) Allocated CPU proportions

(b) Allocated memory proportions

Figure 6: Allocated resource proportions for first fit and FUGA.

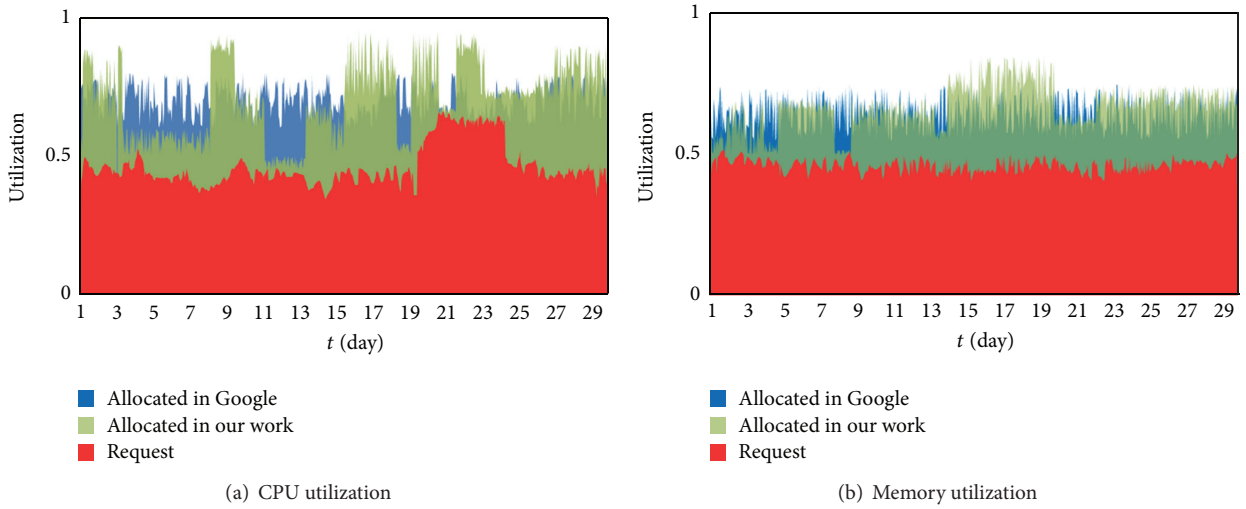
The Google trace provides the information of users' requirements and the actual allocated resources of running tasks in Google cluster [34]. To collect the utilization for CPU and memory, each simulation was divided into every day periods and ran for ten times to take the average value.

Figure 7 shows the utilization simulated in our framework over the 29-day period. In Figure 7(a), overall allocated CPU does not exceed about 75% of the resource capacity. However, FUGA shows that the CPU utilization is over 75% at more than half of the time and sometimes exceeds 85%. Total memory requirements shown in Figure 7(b) are almost account for 50%, and the actual allocated memory in our algorithm is about 10% more than in Google cluster for average, although sometimes it is lower than in Google cluster.

In contrast to the utilization of Google cluster, our algorithm provides more efficient resource allocation decisions. It is not only because FUGA leads to less resource fragments during the allocation as discussed in last section. FUGA also tries to minimize the uneven utilizations for multiple resource dimensions while making the decision on resource allocation.

6. Conclusion

In this paper, we have investigated the resource allocation problem in cloud computing. We consider multiple types of resources like CPU, memory, and storage on virtual machine level to propose an allocation algorithm called FUGA. The algorithm supports not only fair resource allocation for users, but also efficient resource utilization for each physical server. The resource allocation problem is modeled as a finite extensive game with perfect information and the FUGA algorithm results in a Nash equilibrium decision.



(a) CPU utilization

(b) Memory utilization

Figure 7: Resource utilization in Google and FUGA.

Some experiments and simulations are conducted to evaluate the performance of FUGA by comparing to other related works. The results show that the proposed FUGA can achieve better performance in fair allocation than Hadoop scheduler. FUGA can also guarantee more efficient resource allocation rather than the first fit algorithm and the allocation mechanism in Google cluster by setting the proper parameters for the fairness and utilization tradeoff.

Future work could usefully study the fairness-utilization tradeoff when jobs have machine preferences. Another direction involves considering the allocation problem under the job priority situation. Moreover, we plan to investigate how to use this game theoretic resource allocation into a federated environment with multiple resource providers.

Notations

$\{i \mid 1, 2, ..., s\}$: Cloud users
$\{j \mid 1, 2, ..., k\}$: Different kinds of computing resources (CPU, memory, storage, etc.)
$\{m \mid 1, 2, ..., p\}$: Physical servers with idle resources
$C^{(m)}$: Resource capacity vector of physical server m
$\overrightarrow{r_i}$: The VM type required by user i
com: Possible VM combinations of a physical server
R: Resource requirement matrix of all users
$a_{ij}^{(m)}$: The amount of resource j on physical server m allocated to user i
φ_{ij}: Total amount of resource j allocated to user i
A: Resource allocation decision
v: Fairness variance for an allocation decision
η: The number of possible strategies for each player
ske: Unevenness for the utilization of resources
Π: The initial resource space of a physical server
$u^{(m)}$: Resource utilization for physical server m
U: The utility function of resource allocation game.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was partially supported by the NSF of China under Grants no. 61173048 and no. 61300041 and Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant no. 20130074110015.

References

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] S. Son, G. Jung, and S. C. Jun, "An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider," *Journal of Supercomputing*, vol. 64, no. 2, pp. 606–637, 2013.

[3] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, p. 24, Boston, Mass, USA, 2011.

[4] S. Caton and O. Rana, "Towards autonomic management for cloud services based upon volunteered resources," *Concurrency and Computation: Practice & Experience*, vol. 24, no. 9, pp. 992–1014, 2012.

[5] J. Espadas, A. Molina, G. Jiménez, M. Molina, R. Ramírez, and D. Concha, "A tenant-based resource allocation model for scaling Software-as-a-Service applications over cloud computing infrastructures," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 273–286, 2013.

[6] J. O. Gutierrez-Garcia and K. M. Sim, "GA-based cloud resource estimation for agent-based execution of bag-of-tasks applications," *Information Systems Frontiers*, vol. 14, no. 4, pp. 925–951, 2012.

[7] J. Bi, Z. Zhu, R. Tian, and Q. Wang, "Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center," in *Proceedings of the 3rd IEEE International Conference on Cloud Computing (CLOUD '10)*, pp. 370–377, July 2010.

[8] D. C. Vanderster, N. J. Dimopoulos, R. Parra-Hernandez, and R. J. Sobie, "Resource allocation on computational grids using a utility model and the knapsack problem," *Future Generation Computer Systems*, vol. 25, no. 1, pp. 35–50, 2009.

[9] D. Ye and J. Chen, "Non-cooperative games on multidimensional resource allocation," *Future Generation Computer Systems*, vol. 29, no. 6, pp. 1345–1352, 2013.

[10] M. Hassan, B. Song, and E. N. Huh, "Game-based distributed resource allocation in horizontal dynamic cloud federation platform," in *Algorithms and Architectures for Parallel Processing*, Y. Xiang, A. Cuzzocrea, M. Hobbs, and W. Zhou, Eds., vol. 7016 of *Lecture Notes in Computer Science*, pp. 194–205, Springer, 2011.

[11] "Scheduling in Hadoop," 2012, http://www.cloudera.com/blog/tag/scheduling.

[12] C. A. Waldspurger, *Lottery and Stride Scheduling: Flexible Proportional-Share Resource Management*, Massachusetts Institute of Technology, 1995.

[13] T. Lan, D. Kao, M. Chiang, and A. Sabharwal, "An axiomatic theory of fairness in network resource allocation," in *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '10)*, pp. 1–9, March 2010.

[14] D. C. Parkes, A. D. Procaccia, and N. Shah, "Beyond dominant resource fairness: extensions, limitations, and indivisibilities," in *Proceedings of the 13th ACM Conference on Electronic Commerce*, pp. 808–825, Valencia, Spain, 2012.

[15] X. Wang, X. Liu, L. Fan, and X. Jia, "A decentralized virtual machine migration approach of data centers for cloud computing," *Mathematical Problems in Engineering*, vol. 2013, Article ID 878542, 10 pages, 2013.

[16] D. C. Erdil, "Autonomic cloud resource sharing for intercloud federations," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1700–1708, 2013.

[17] M. Steinder, I. Whalley, D. Carrera, I. Gaweda, and D. Chess, "Server virtualization in autonomic management of heterogeneous workloads," in *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM '07)*, pp. 139–148, May 2007.

[18] S. Di and C. L. Wang, "Dynamic optimization of multiattribute resource allocation in self-organizing clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 3, pp. 464–478, 2013.

[19] M. Cardosa, A. Singh, H. Pucha, and A. Chandra, "Exploiting spatio-temporal tradeoffs for energy-aware MapReduce in the cloud," *IEEE Transactions on Computers*, vol. 61, no. 12, pp. 1737–1751, 2012.

[20] P. Jord, C. Castillo, D. Carrera, Y. Becerra, I. Whalley et al., "Resource-aware adaptive scheduling for mapreduce clusters," in *Proceedings of the 12th ACM/IFIP/USENIX International Conference on Middleware*, pp. 187–207, Lisbon, Portugal, 2011.

[21] T. Sandholm and K. Lai, "MapReduce optimization using regulated dynamic prioritization," in *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '09)*, pp. 299–310, Seattle, Wash, USA, June 2009.

[22] M. Zukerman, L. Tan, H. Wang, and I. Ouveysi, "Efficiency-fairness tradeoff in telecommunications networks," *IEEE Communications Letters*, vol. 9, no. 7, pp. 643–645, 2005.

[23] C. Joe-Wong, S. Sen, L. Tian, and C. Mung, "Multi-resource allocation: fairness-efficiency tradeoffs in a unifying framework," in *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '12)*, pp. 1206–1214, 2012.

[24] Z. Xiao, W. J. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, 2013.

[25] N. Nisan, *Algorithmic Game Theory*, Cambridge University Press, 2007.

[26] M. J. Osborne, *An Introduction to Game Theory*, vol. 3, Oxford University Press, New York, NY, USA, 2004.

[27] Y. Shoham, "Computer science and game theory," *Communications of the ACM*, vol. 51, no. 8, pp. 75–79, 2008.

[28] L. Guijarro, V. Pla, J. R. Vidal, and J. Martinez-Bauset, "Entry, competition, and regulation in cognitive radio scenarios: a simple game theory model," *Mathematical Problems in Engineering*, vol. 2012, Article ID 620972, 13 pages, 2012.

[29] A. Iqbal and A. H. Toor, "Quantum mechanics gives stability to a Nash equilibrium," *Physical Review A*, vol. 65, Article ID 022306, 5 pages, 2002.

[30] P. J. Reny, "Backward induction, normal form perfection and explicable equilibria," *Econometrica*, vol. 60, no. 3, pp. 627–649, 1992.

[31] F. Schuhmacher, "Proper rationalizability and backward induction," *International Journal of Game Theory*, vol. 28, no. 4, pp. 599–615, 1999.

[32] H. Xu and B. C. Li, "Anchor: a versatile and efficient framework for resource management in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1066–1076, 2013.

[33] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *Proceedings of the 7th International Conference On Autonomic Computing*, pp. 11–20, June 2010.

[34] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: google trace analysis," in *Proceedings of the 3rd ACM Symposium on Cloud Computing*, pp. 1–13, San Jose, Calif, USA, 2012.