Imperfect Information Dynamic Stackelberg Game Based Resource Allocation Using Hidden Markov for Cloud Computing

Wei Wei; Xunli Fan; Houbing Song, *Senior Member, IEEE*; Xiumei Fan; and Jiachen Yang

Abstract—Existing static grid resource scheduling algorithms, which are limited to minimizing the makespan, cannot meet the needs of resource scheduling required by cloud computing. Current cloud infrastructure solutions provide operational support at the level of resource infrastructure only. When hardware resources form the virtual resource pool, virtual machines are deployed for use transparently. Considering the competing characteristics of multi-tenant environments in cloud computing, this paper proposes a cloud resource allocation model based on an imperfect information Stackelberg game (CSAM-IISG) using a hidden Markov model (HMM) in a cloud computing environment. CSAM-IISG was shown to increase the profit of both the resource supplier and the applicant. Firstly, we used the HMM to predict the service provider's current bid using the historical resources based on demand. Through predicting the bid dynamically, an imperfect information Stackelberg game (IISG) was established. The IISG motivates service providers to choose the optimal bidding strategy according to the overall utility, achieving maximum profits. Based on the unit prices of different types of resources, a resource allocation model is proposed to guarantee optimal gains for the infrastructure supplier. The proposed resource allocation model can support synchronous allocation for both multi-service providers and various resources. The simulation results demonstrated that the predicted price was close to the actual transaction price, which was lower than the actual value in the game model. The proposed model was shown to increase the profits of service providers and infrastructure suppliers simultaneously.

Index Terms—Cloud computing, game theory, hidden Markov model, resource allocation

1 Introduction

ONE of the most important challenges of cloud computing is how to make a reasonable application migration to reduce energy consumption.

A key problem in cloud computing is resource scheduling. Although many static resource scheduling algorithms have been proposed for grid computing, those algorithms cannot be used for resource scheduling in cloud computing for the following three reasons:

- Grid computing resource scheduling achieves optimal allocation of resources, but the time required for the virtual machine application and release in cloud computing makes a static resource scheduling algorithm unsuitable for cloud computing.
- The running time of a grid task is determined by the static resource scheduling strategy, but in cloud computing the use of the virtual machine is determined by the user; i.e., resource scheduling in cloud computing cannot determine the use of virtual machines.
- The purpose of resource scheduling in grid computing is to reduce the running time of the task, but the main purpose of resource scheduling in cloud computing is to reduce data center power consumption, because the use of the virtual machine in cloud computing is unchanged.

Cloud computing enables users to get services through network and computing resources on-demand directly. Since Amazon proposed the concept of cloud computing, many cloud computing systems, including Amazon EC2, the Google App Engine, Apache Hadoop, and Microsoft Azure, have been developed [1]. Amazon EC2 is Amazon's elastic computing cloud. The web service is actually provided by a Linux virtual machine resource, also called an instance, in the Amazon data center. Depending on their sizes, instances can be divided into three types: small, large, and very large. The Google App Engine is a cloud computing platform that provides users with a certain resource freely, charging a fee only when the user exceeds baseline CPU, storage space, bandwidth, and other resources. Apache Hadoop is a software system that supports data intensive distributed applications, similar to the MapReduce framework of Google and the Google file system. Hadoop consists of three parts: the Hadoop distributed file system (HDFS), HBase, and MapReduce.

- *W. Wei is with the School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, Shaanxi 710048 China. E-mail: weiwei@xaut.edu.cn.*
- *X. Fan is with the School of Information Science and Technology, Northwest University, Xi'an, Shaanxi 710027 China. E-mail: xunlfan@nwu.edu.cn.*
- *H. Song is with the Department of Electrical and Computer Engineering, West Virginia University, Montgomery, WV 25136. E-mail: Houbing.Song@mail.wvu.edu (Corresponding Author)*
- *X. Fan is with the School of Automation and Information Engineering, Xi'an University of Technology, Xi'an, Shaanxi 710048 China. E-mail: xmfan@xaut.edu.cn.*
- *J. Yang is with the School of Electronic Engineering, Tianjin University, Tianjin 300072 China. E-mail: yangjiachen@ tju.edu.cn.*

xxxx-xxxx/0x/$xx.00 © 200x IEEE Published by the IEEE Computer Society

The HDFS creates a number of copies of data. MapReduce divides a task into multiple subtasks, each of which is performed by a computing node. Microsoft Azure is a platform providing a variety of on-demand services, including Windows Azure, Azure SOL, and Windows Azure AppFabric. IBM has also proposed a cloud computing program called Blue Cloud.

The biggest advantage of cloud computing is resource allocation, which can maximize resource utilization and minimize operating costs. Flexibility derives from virtualization technology, including hardware virtualization such as CPU, memory, storage, and network [2]. Virtualization technology provides virtualized computing, storage, and networks for upper layer users, services, and applications, and assigns resources dynamically according to the user's demand. For market operations, virtualization technology leads to macro changes in the operation framework. Zhang et al. [3] proposed a migration decision strategy that decides whether to perform an application migration based on different applications of energy consumption in the cloud.

From the perspective of services, operations are clearly divided into infrastructure suppliers (INs) and service providers (SP) [4]. The main responsibility of INs is to deploy and manage underlying network resources, and to provide SPs with different resources through a programming interface. The IN does not directly provide services to users. Server virtualization is the underlying core support technology of cloud computing; it generates multiple virtual servers for users by operating system virtualization of the physical server.

Virtualization constructs and encapsulates the resource capabilities of computing, storage, networks, and other resources into a resource pool that provides on-demand services. The main duty of SPs is to rent the bottom network resources of INs, create a virtual network, and provide professional services to end users [5].

How to rent virtual resources efficiently and reasonably is one of the key issues in cloud computing. Virtual resources attracted more attention after virtualization technology matured [6]. The existing research in virtual resource management can be classified into two categories: virtual resource deployment and rent transaction. Virtual resource deployment studies how to allocate limited hardware resources to meet the needs of more users, and how to create a virtual network for a user directly [7]. Tenant transaction studies the supply-demand relationship between INs and SPs in the multi-tenancy cloud market competition environment. The goal of tenant transaction is to maximize overall societal interests and ensure a fair, efficient competitive environment. Existing research focuses on simulating the resources leasing market in a cloud environment and learning from auction theory in the field of economics. Participants are motivated to choose a reasonable transaction price to get a balanced strategy with optimal overall interest.

However, in a cloud environment the resources, such as CPU, storage, and bandwidth, are diversified. Multidimensional resource auction issues, especially the margin of standard combinatorial auctions, are NP-hard problems [8]. A multidimensional resource auction is inefficient in the large solution space at present and needs to be solved using a heuristic algorithm. In addition, an auction scheme using a uniform resource price determines the resource's average unit price by simply dividing all the resources according to the user demand bid. However, the problem with auction schemes is their lack of incentives and fairness.

To solve resource allocation in the cloud computing environment, this paper proposes a cloud resource allocation model based on an imperfect information Stackelberg game model (CSAM-IISG) using the hidden Markov model (HMM). SPs compete in the multi-tenant cloud market environment. First, we use an HMM to predict a current price based on the SP's historical resource demand. Then, we construct an imperfect information Stackelberg game model (IISG) by dynamically predicting the bid. At the same time, IISG can motivate SPs to choose the optimal bidding strategy in line with the overall interest of the purchase in order to achieve the maximum profit. Finally, a resource allocation model based on the unit price of different types of resources is designed to maximally benefit INs.

The rest of this paper is organized as follows. Section 2 describes the current situation of cloud resource allocation and its drawbacks. Section 3 describes the prediction model of the SP's bid based on an HMM. Section 4 presents the proposed dynamic Stackelberg game price model among SPs using the Cobb-Douglas utility function. Section 5 describes the design of the resource allocation model based on different resource prices. Section 6 presents the performance evaluation results of the proposed algorithm. Section 7 concludes the paper.

2 State of the Art of Cloud Resource Allocation

The resource allocation problem is an important challenge in a large-scale, distributed cloud computing environment. How to allocate resources reasonably and efficiently has been a key issue in recent years. Cloud computing is evolved from grid computing. However, existing resource allocation methods in grid computing cannot be used directly in the cloud environment because user demands in the cloud are greater, dynamic, and diverse [9]. Therefore, the problem of low utilization of cloud resources, especially the data centers of cloud infrastructure, has been a focus of academia and industry [10].

Cloud resource allocation problems can be classified into two categories. One category is maximization of resource utilization. According to a user's specific demands—for example, CPU, bandwidth, and storage—hardware resources of INs are assigned as many as possible to improve resource utilization [11]. The other category is maximization of profit, which may be the overall profit or the profit of INs [12].

Considering the cloud market operating mechanism, resource allocation between SPs and INs should be more focused on economic profits, while resource allocation between the SPs and users should be more focused on

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2528246, IEEE Transactions on Services Computing

efficiency.

Zaheer et al. [13] proposed an auction framework based on fair resource allocation, which minimized the cost to a single SP but failed to obtain global optimal resource allocation in a multi-SP competitive environment. Trinh et al. [14] proposed a game theory strategy to solve resource allocation that considered bandwidth allocation but did not consider the limited resources provided by INs. [15] proposed a game-theoretic approach to energy-efficient resource allocation in device-to-device underlay communications. [16] and [17] addressed multilevel authentication issue and multicloud issue in cloud computing, respectively. Based on a Vickrey auction mechanism, [18] proposed a dynamic cloud resource auction mechanism where each user submits their required resource bids, and the highest bidder wins the auction. This model maximized the benefits of resource providers. However, there was only one winner each time, and the efficiency of allocation was low. Combining two-way auction theory from a service management perspective, [19] proposed a virtual network resource allocation model for multi-INs and a multi-SP competition model. However, it could not obtain the optimal solution among resource prices with large differences. Based on a random integer programming, [20] presented an optimized cloud resource allocation model in which the overall allocation was split into several sub-problems to increase computational efficiency. However, this was a centralized algorithm, which is inefficient for large users. As in traditional resource allocation, it ignored the user's profit, because the resource provider pricing mechanism only concerned the interests of resource providers [21]. Dynamic pricing mechanisms can increase user benefits, increase the competitive health of resource providers, and improve cloud resource utilization [22, 28-30].

The drawbacks of the above studies can be summarized as follows:

- Most studies addressed only how to create a fair trade environment and get higher economic benefits and effectiveness in the case of a single SP. The efficiency is very low when the number of users is large.
- Most algorithms considered only a single resource allocation, but in practice resource types are very diverse and there are differences among evaluation prices and the number of SPs in cloud environments. A resource allocation algorithm that allocates multiple resources simultaneously is more in line with actual needs.
- Most existing multiple INS resource allocation techniques are based on combinatorial auctions, which improve the fairness of market competition and maximize profit, but do not consider the non-cooperative behavior of SPs. In addition, combinatorial auctions lack fairness.
- The use of a dynamic pricing mechanism in resource allocation is helpful to build more rational and efficient mechanisms, but the efficiency of centralized pricing is very low.

To overcome the above drawbacks, this paper proposes a cloud resource allocation model called CSAM-IISG to improve the profits of both the resource supply and the resource on demand.

3 SP Resource Bid Forecast Based on the HMM

3.1 Task Migration Model

Fig. 1 shows the application model of cloud migration. The application consists of n linear topological structures of the task; the completion time of the entire application is limited to T_d. For task k, its computational load is w_k, the amount of data input is α_k, and the output data is β_k. The output data of task *k-1* is the input data of task k. In the start state S, when it receives the output data β_n of the task n, the terminal enters the end state D at the end of the application.

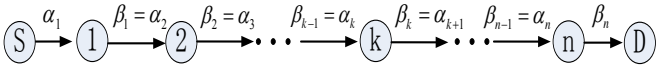
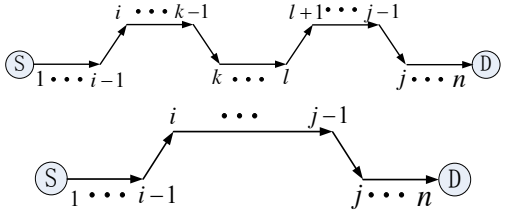


Figure 1. Application task model

If both the adjacent two tasks are executed in the cloud, we can get the strategy feature of the minimum execution time, with the theorem as follows:

Theorem 1: In the Markov random model, for any migration strategy $p \in P$, if p can enable the smallest execution time of the applications, then the number of migration p will not be greater than 1.



(a) Two migrations (b) One migration

Figure 2. Migration mode

Proof: After the terminal generates data α_1 at the state S, the application begins to execute; after receiving the output data β_n of task n, the terminal enters the end state D at the end of the application. Therefore, the application starting at state S and ending at state D must be executing on a terminal. If the time of task 1 performed on the terminal is less than its upload time on output data, the task will be arranged on a terminal in such a way as to reduce the execution time. The performance time of task 1 and task 2 on the terminal is less than the upload time of task 2 on output data. Tasks 1 and 2 will be arranged on the terminal in such a way as to reduce execution time. If the time of task n on the terminal performed on the download time of its input data and output data is less than its download time, then task n is arranged on the terminal in such a way as to reduce execution time. Based on the above assumptions, task i is the first task migrated to the cloud to perform, and task j is the last task returned to the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2528246, IEEE Transactions on Services Computing

terminal to perform.

[End of proof]

Inference 1: In the Markov random model, for any migration policy, if the resource consumption of the terminal is minimized, the number of migrations is not greater than 1.

Theorem 2: In the Markov random model, assuming that task *i-1* has been completed on the terminal, the state of the current time slot *t* is g_t. If we migrate task *i* to perform in the cloud, we can minimize the resource consumption of terminal, task *i* must satisfy:

$$\max[0,\frac{\alpha_i - R_t}{w_i \mathrm{E}[R]}] - \frac{\beta_i}{w_i \mathrm{E}[R]} + \frac{1}{w_i} < \frac{1}{P_s}(\frac{P_m}{f_m} - \frac{P_i}{f_c}) \quad (1)$$

Proof: Supposing that task *i-1* has been executed on the terminal, and task *i* is migrated to perform in the cloud in the current time slot *t*, then:

$$E_i = \sum_{k=1}^{i-1}\frac{w_k}{f_m}P_m + \mathrm{E}(d_s(i))P_s + \sum_{k=i}^{j-1}\frac{w_k}{f_c}P_i + \mathrm{E}(d_r(j-1))P_r + \sum_{k=j}^{n}\frac{w_k}{f_m}P_m$$

$$= \begin{cases} \sum_{k=1}^{i-1}\frac{w_k}{f_m}P_m + (1+\max[0,\frac{\alpha_i - R_G}{\mathrm{E}(R)}])P_s + \sum_{k=i}^{j-1}\frac{w_k}{f_c}P_i + \frac{\beta_{j-1}}{\mathrm{E}[R]}P_r + \sum_{k=j}^{n}\frac{w_k}{f_m}P_m, & R_t = R_G \\ \sum_{k=1}^{i-1}\frac{w_k}{f_m}P_m + (1+\max[0,\frac{\alpha_i - R_B}{\mathrm{E}(R)}])P_s + \sum_{k=i}^{j-1}\frac{w_k}{f_c}P_i + \frac{\beta_{j-1}}{\mathrm{E}[R]}P_r + \sum_{k=j}^{n}\frac{w_k}{f_m}P_m, & R_t = R_B \end{cases} \quad (2)$$

Assuming the postponed task *i+1* is migrated to perform in the cloud, at the current time slot *t*, then:

$$E_i^{'} = \sum_{k=1}^{i}\frac{w_k}{f_m}P_m + \frac{\alpha_{i+1}}{\mathrm{E}[R]}P_s + \sum_{k=i+1}^{j-1}\frac{w_k}{f_c}P_i + \frac{\beta_{j-1}}{\mathrm{E}[R]}P_r + \sum_{k=j}^{n}\frac{w_k}{f_m}P_m$$

If the *i*-th execution task is migrated to perform in the cloud, the resource consumption can be minimized, then $E_i < E_i^{'}$, when $R_t = R_G$, then:

$$E_i - E_i^{'} = \frac{w_i}{f_c}P_i - \frac{w_i}{f_m}P_m + (1+\max[0,\frac{\alpha_i - R_G}{\mathrm{E}(R)}])P_s - \frac{\alpha_{i+1}}{\mathrm{E}(R)}P_s < 0 \quad (3)$$

Because $\alpha_{i+1} = \beta_i$, according to Equation 3, we obtain:

$$\max[0,\frac{\alpha_i - R_G}{w_i \mathrm{E}[R]}] - \frac{\beta_i}{w_i \mathrm{E}[R]} + \frac{1}{w_i} < \frac{1}{P_s}(\frac{P_m}{f_m} - \frac{P_i}{f_c}) \quad (4)$$

If $R_t = R_B$, according to Equation 3 we can get the following:

$$\max[0,\frac{\alpha_i - R_B}{w_i \mathrm{E}[R]}] - \frac{\beta_i}{w_i \mathrm{E}[R]} + \frac{1}{w_i} < \frac{1}{P_s}(\frac{P_m}{f_m} - \frac{P_i}{f_c})[\quad (5)$$

By Equations 4 and 5, we get Formula 1.

[End of proof]

The implementation process of the Markov model based migrated algorithm is shown as follows:

Step 1: Initialize the group $I^{(0)}$.

Step 2: Add the task *1,2,…, n* executed on a terminal to the initial population.

Step 3: Add the new members randomly generated to the parent population.

Step 4: Determine whether the resource consumption is less than f_{BEP}. If yes, add it to the progeny groups $I^{(i)}$, else give up.

Step 5: Determine whether it meets the stop condition. If yes, output the best individual, otherwise go to step 2.

In practice, the local problem is the major problem. To avoid the local problem, based on a simple genetic algorithm, and according to Theorems 1 and 2, we considered the following elements:

- In the initial population, members were added to perform on the terminal. To ensure that in the worst-case conditions, there was migration policy output.
- Some new members were plus in each cycle, which stratified the following conditions: when *i<j*, if the tasks *i-1* and *j* were performed on the terminal, and the tasks *i* and *j-1* were executed in the cloud, then the tasks *1,2,…,i-1,j, j+1,…n* were executed in the terminal, and tasks *i,i+1, …,j-1* were performed in the clouds to improve the diversity of the population.
- In the crossover phase, if the tasks *i,j(i<j)* were executed in the cloud, the individual sub-tasks *i,i+1,…,j* were performed in the cloud.
- The fitness function was set to $f_{\mathrm{BPP}} = \min[e_m(\mathrm{A}), e_c(\mathrm{A})]$, where $e_m(\mathrm{A}) = \sum_{k=1}^{n} e_m(k)$, $e_c(\mathrm{A}) = e_s(1) + \sum_{k=1}^{n} e_c(k) + e_r(n)$.

3.2 Bid Forecast Based on the HMM

In a multi-tenant market business model in cloud environments, the SPs are independent and competitive. They cannot expose all their information to competitors. In order to build a rapid and flexible dynamic price game, we used the HMM [23] to predict SP bids.

We used the SP's historical resource request sequence to predict the bid of the SP's next resource allocation and forecast the SP's resource bid through the HMM. The structure of the HMM is shown in Fig. 3.

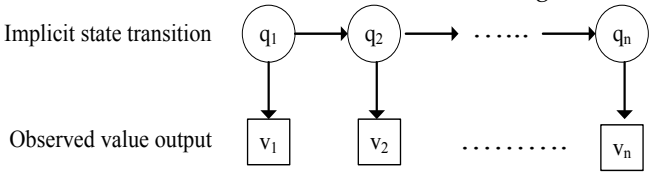


Figure 3. The structure of the HMM

The hidden Markov model was described by a 5-tuple: $<\Omega_X, \Omega_O, \pi, \mathrm{A}, \mathrm{B}>$. The HMM included two state sets and three probability matrixes.

- $\Omega_X = \{q_1, \ldots, q_N\}$ was a finite set of states.
- $\Omega_O = \{v_1, \ldots, v_M\}$ was a finite set of observed value.
- $A = \{a_{ij}\}$, $a_{ij} = p(X_{t+1}=q_j \mid X_t=q_i)$ was the transition probability.
- $B=\{b_{ik}\}$, $b_{ik} = p(O_t=v_k \mid X_t=q_i)$ was the output probability.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2528246, IEEE Transactions on Services Computing

- $\pi=\{\pi_i\}$, $\pi_i=p(X_1=q_i)$ was the initial state distribution.

For virtual network resource allocation,

- $\Omega_X=\{q_1,q_2,.....,q_m\}$ presented the set of resource unit bids offered by an SP.
- $\Omega_O=\{v_1,v_2,.....,v_n\}$ was the historical resource transaction record of any SP at time t.
- $\pi=p(X_1=q_i)\ (1\le i\le m)$ was the probability distribution of the initial bid offered by the SP.
- $A=\{a_{q_iq_j}\},a_{q_iq_j}=p(X_{t+1}=q_j\mid X_t=q_i)\ (1\le i\le m,1\le j\le m)$ was the transfer matrix of various states in Ω_X.
- $B=\{b_{q_{ig}v_i}\},b_{q_{ig}v_i}=p(O_t=v_i\mid X_t=q_g)\ (1\le i\le n,1\le g\le m)$ was the observed probability of resource request v_i at the possible bid of q_g.

Since Ω_O and Ω_X were two independent events and $p(O_t=v_i)>0$, according to the definition of conditional probability we obtained Equation 6.

$$p(X_{t+1}=q_j\mid O_{t+1}=v_{t+1})=\frac{p\{X_{t+1}=q_j,O_{t+1}=v_{t+1}\}}{p(O_{t+1}=v_{t+1})} \quad (6)$$

According to the multiplication formula, we obtained Equation 7.

$$p\{X_{t+1}=q_j,O_{t+1}=v_{t+1}\}=p(O_{t+1}=v_{t+1})p(X_{t+1}=q_j\mid O_{t+1}=v_{t+1}) \quad (7)$$

Let $p\{X_{t+1}=q_j,O_{t+1}=v_{t+1}\}=\alpha_{t+1}(q_j,v_{t+1})$, then, using the total probability formula, we obtained Equation 8.

$$\begin{aligned}\alpha_{t+1}(q_j,v_{t+1})&=p\{q_j\mid v_{t+1}\}\sum_{q_i\in X}\alpha_{t+1}(q_j,v_{t+1})\\&=\sum_{q_i\in X}p\{X_t=q_i,O_t=v_t,X_{t+1}=q_j,O_{t+1}=v_{t+1}\}\\&=\sum_{q_i\in X}\alpha_t(q_i,v_t)p\{X_{t+1}=q_j\mid X_t=q_i\}p\{O_{t+1}=v_{t+1}\mid X_{t+1}=q_j\}\end{aligned} \quad (8)$$

Since $p(X_{t+1}=q_j\mid X_t=q_i)=a_{q_iq_j}$, $p(O_{t+1}=v_{t+1}\mid X_t=q_i)=b_{q_jv_{t+1}}$, substituting it into Equation 8 we obtained Equation 9.

$$\alpha_{t+1}(q_j,v_{t+1})=\sum_{q_i\in X}\alpha_t(q_i,v_t)a_{q_iq_j}b_{q_iv_{t+1}} \quad (9)$$

Then, $\alpha_{t+1}(q_j,v_{t+1})$ could be calculated using Equation 9.

When substituting $\alpha_{t+1}(q_j,v_{t+1})$ into Equation 6, we could obtain the probability of all the possible bids offered by the SPs and select the maximum probability q_j to bid.

At present, no research describes how to apply the HMM to cloud resource allocation. This study applied the HMM to predict SP bids, and constructed the imperfect information dynamic Stackelberg game model based on a price prediction. This determined the price quickly and flexibly in accordance with the current market state, achieved optimized benefits, and obtained the Nash equilibrium.

4 The SP Dynamic Price Strategy

Stackelberg is a model of imperfect competition based on a non-cooperative game. In game theory, a Stackelberg model is a sequential game in which there are two firms, which sell homogeneous products, and are subject to the same demand and cost functions. One firm, the leader, is perhaps better known or has greater brand equity, and is therefore better placed to decide first which quantity to sell, and the other firm, the follower, observes this and decides on its production quantity.

4.1 The Cobb-Douglas Utility Function

The Cobb–Douglas production function is a particular functional form of the production function, widely used to represent the technological relationship between the amounts of two or more inputs, particularly physical capital and labor, and the amount of output that can be produced by those inputs. In economics, the Cobb–Douglas utility function [24] is widely used to represent the technological relationship between the amounts of two or more inputs. In the most standard form for production of a single good with two factors, the function is $Y=AL^{\beta}K^{\alpha}$, where Y is the total production, L is the labor input, K is the capital input, A is the total factor productivity, and α and β are the output elasticity of capital and labor respectively. Assuming perfect competition and $\alpha+\beta=1$, α and β can be shown to be capital's and labor's shares of output.

Furthermore, if $\alpha+\beta=1$, the production function has constant returns to scale, which means that doubling the usage of capital K and labor L also doubles output Y. If $\alpha+\beta<1$, returns to scale are decreasing, and if $\alpha+\beta>1$, returns to scale are increasing. Cobb and Douglas were influenced by statistical evidence that appeared to show that labor and capital shares of total output were constant over time in developed countries; they explained this by statistical fitting least-squares regression of their production function.

The Cobb–Douglas production function was not developed based on any knowledge of engineering, technology, or management of the production process. It was instead developed because it had attractive mathematical characteristics, such as diminishing marginal returns to either factor of production and the property that expenditure on any given input is a constant fraction of total cost. A major criticism at the time was that estimates of the production function, although seemingly accurate, were based on such sparse data that it was hard to give them much credibility. Many modern researchers develop models that give Cobb–Douglas production function from the micro level; it is nevertheless a mathematical mistake to assume that just because the Cobb–Douglas function applies at the micro-level, it also always applies at the macro-level. Similarly, it is not necessarily the case that a macro Cobb–Douglas applies at the disaggregated level.

4.2 The SP's Utility Function

Cloud computing virtualizes the different hardware resources, such as CPU, storage, and memory, in a data center by virtualization technology. When the hardware resources form the virtual resource pool, virtual machines are deployed to be used transparently for users. Job scheduling and resource scheduling are two key technol-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2528246, IEEE Transactions on Services Computing

ogies in cloud computing. Increases in the size of data centers and number of users increases cloud virtual machine efficiency and enables tasks to finish rapidly—an important issue in cloud resource scheduling. By analyzing the characteristics of cloud resources, we summarized the model, the goal, and the characteristics of virtual machine scheduling. When an SP obtains the resource predicted price of CPU, bandwidth, and storage requested by other SPs using the HMM, their own stable equilibrium bid could be obtained using the dynamic Stackelberg pricing game in this paper. In dynamic game price strategy, the SP determines the bidding strategies and maximizes the Cobb-Douglas utility function.

$$U_i=(m_i^{cpu}-s_i^{cpu})^{\alpha}\,\mathrm{P}\,s_i^{cpu}(\mathrm{w})^{\beta}+(m_i^{bw}-s_i^{bw})^{\alpha}\,\mathrm{P}\,s_i^{bw}(\mathrm{w})^{\beta}+(m_i^{\mathrm{storage}}-s_i^{\mathrm{storage}})^{\alpha}\,\mathrm{P}\,s_i^{\mathrm{storage}}(\mathrm{w})^{\beta} \tag{10}$$

where $m_i^{cpu}, m_i^{bw}, m_i^{\mathrm{storage}}$ were actual valuations of the CPU, bandwidth, and storage; $Ps_i^{cpu}, Ps_i^{bw}, \mathrm{P}s_i^{\mathrm{storage}}$ were the possible bids of the CPU, bandwidth, and storage offered by the SP; and $s_i^{cpu}=q_i^{cpu}v_i^{cpu}$, $s_i^{bw}=q_i^{bw}v_i^{bw}$, $s_i^{\mathrm{storage}}=q_i^{storage}v_i^{storage}$, $\mathrm{P}\,s_i^{cpu}(\mathrm{w})$, $\mathrm{P}\,s_i^{bw}(\mathrm{w})$, $\mathrm{P}\,s_i^{\mathrm{storage}}(\mathrm{w})$ were the winning probability of the bid by the SP. The size of α and β reflected the emphasis on profits and the winning probability [21]. Equation 10 was equivalent to Equation 11

$$U_i=(m_i^{cpu}-s_i^{cpu})^{\lambda}\,\mathrm{P}\,s_i^{cpu}(\mathrm{w})+(m_i^{bw}-s_i^{bw})^{\lambda}\,\mathrm{P}\,s_i^{bw}(\mathrm{w})+(m_i^{\mathrm{storage}}-s_i^{\mathrm{storage}})^{\lambda}\,\mathrm{P}\,s_i^{\mathrm{storage}}(\mathrm{w}),\ \lambda=\alpha/\beta>0,0<\alpha\le 1,0<\beta\le 1,\alpha+\beta=1 \tag{11}$$

where parameters α,β were the weights giving relative importance to the remaining resources and the average remaining time, and $\alpha+\beta=1$. $\alpha=1$ meant that only the remaining resource constraint was considered in the final bid value; $\beta=1$ meant that only the remaining time constraint was considered; and $0<\alpha\le 1,0<\beta\le 1$ meant that both constraints were taken into account.

This paper assumed that if the SP had the same emphasis on profits as on the winning probability, that is $\alpha=\beta$, then Equation 11 could be simplified to Equation 12.

$$U_i=(m_i^{cpu}-s_i^{cpu})\,\mathrm{P}\,s_i^{cpu}(\mathrm{w})+(m_i^{bw}-s_i^{bw})\,\mathrm{P}\,s_i^{bw}(\mathrm{w})+(m_i^{\mathrm{storage}}-s_i^{\mathrm{storage}})\,\mathrm{P}\,s_i^{\mathrm{storage}}(\mathrm{w}) \tag{12}$$

The higher unit resource price offered by the SP, the greater the probability of winning. Then

$$\mathrm{P}\,s_i^{cpu}(\mathrm{w})=\left(\frac{s_i^{cpu}}{v_i^{cpu}}\right)\Big/\left(\frac{s_i^{cpu}}{v_i^{cpu}}+\sum_{j=1}^{\kappa-1}q_j^{cpu}\right) \tag{13}$$

$$\mathrm{P}\,s_i^{bw}(\mathrm{w})=\left(\frac{s_i^{bw}}{v_i^{bw}}\right)\Big/\left(\frac{s_i^{bw}}{v_i^{bw}}+\sum_{j=1}^{\kappa-1}q_j^{bw}\right) \tag{14}$$

$$\mathrm{P}\,s_i^{\mathrm{storage}}(\mathrm{w})=\left(\frac{s_i^{\mathrm{storage}}}{v_i^{\mathrm{storage}}}\right)\Big/\left(\frac{s_i^{\mathrm{storage}}}{v_i^{\mathrm{storage}}}+\sum_{j=1}^{\kappa-1}q_j^{\mathrm{storage}}\right) \tag{15}$$

where $q_j^{cpu}, q_j^{bw}, q_j^{\mathrm{storage}}$ represented the predicted prices of the CPU, bandwidth, and storage respectively. The current SP calculated through other SPs' historical resource request records.

In the SP's Stackelberg game, any SP i 's dominant strategy was the bidding strategy $s_i^*=(s_i^{cpu*}, s_i^{bw*}, s_i^{storage*})$, which maximized the profit function u_i, which was called the Nash equilibrium s_i^*, that is the final bid of SP i. Here is the process of solving the Nash equilibrium:

Substituting Equations 13, 14, and 15 into Equation 12, we obtained the expected profits of any SP. The utility function is shown in Equation 16.

$$U_i=(m_i^{cpu}-s_i^{cpu})\left(\frac{s_i^{cpu}}{v_i^{cpu}}\Big/\left(\frac{s_i^{cpu}}{v_i^{cpu}}+\sum_{j=1}^{\kappa-1}q_j^{cpu}\right)\right)+(m_i^{bw}-s_i^{bw})\left(\frac{s_i^{bw}}{v_i^{bw}}\Big/\left(\frac{s_i^{bw}}{v_i^{bw}}+\sum_{j=1}^{\kappa-1}q_j^{bw}\right)\right)+(m_i^{\mathrm{storage}}-s_i^{\mathrm{storage}})\left(\frac{s_i^{\mathrm{storage}}}{v_i^{\mathrm{storage}}}\Big/\left(\frac{s_i^{\mathrm{storage}}}{v_i^{\mathrm{storage}}}+\sum_{j=1}^{\kappa-1}q_j^{\mathrm{storage}}\right)\right) \tag{16}$$

Equation 16 was simplified to Equation 17.

$$U_i=(m_i^{cpu}-s_i^{cpu})\frac{s_i^{cpu}}{s_i^{cpu}+v_i^{cpu}\sum_{j=1}^{\kappa-1}q_j^{cpu}}+(m_i^{bw}-s_i^{bw})\frac{s_i^{bw}}{s_i^{bw}+v_i^{bw}\sum_{j=1}^{\kappa-1}q_j^{bw}}+(m_i^{\mathrm{storage}}-s_i^{\mathrm{storage}})\frac{s_i^{storage}}{s_i^{storage}+v_i^{storage}\sum_{j=1}^{\kappa-1}q_j^{\mathrm{storage}}} \tag{17}$$

Since $v_i^{cpu}\sum_{j=1}^{\kappa-1}q_j^{cpu}$ is constant, let $v_i^{cpu}\sum_{j=1}^{\kappa-1}q_j^{cpu}=G^{cpu}$, $v_i^{storage}\sum_{j=1}^{\kappa-1}q_j^{storage}=G^{storage}$, $v_i^{bw}\sum_{j=1}^{\kappa-1}q_j^{bw}=G^{bw}$.

4.3 The Optimization Problem of the SP's Utility Function

Theorem 3: If the bid strategy set that the SP bids for the product is a finite positive integer set and is greater than 1, then the expected profit of each SP can achieve the optimal price on a reasonable strategy set, and the system exists in a Nash equilibrium. The constraints can be further relaxed when the following conditions are satisfied:

$$\begin{cases} m_i^{cpu}G^{cpu}+G^{cpu^2}>1 \\ m_i^{bw}G^{bw}+G^{bw^2}>1 \\ m_i^{storage}G^{storage}+G^{storage^2}>1 \end{cases} \tag{18}$$

Proof: In the proposed Stackelberg game model, the number of SPs is limited and the strategy set for each participant is a finite convex set. By the Nash equilibrium existence theorem [28], at least Nash equilibrium $\mathbf{s}_i^*$ is included. The problem of solving Nash equilibrium $\mathbf{s}_i^*$ can be converted into solving the extreme problem of a ternary function U_i. Under the indefinite conditions, we solve the stagnation function U_i and seek the first-order partial derivatives of U_i. We obtain the following first-order partial derivatives of U_i.

$$U'_{i_{s_i^{cpu}}}=\frac{m_i^{cpu}G^{cpu}-(s_i^{cpu})^2-2s_i^{cpu}G^{cpu}}{(s_i^{cpu}+G^{cpu})^2},$$

$$U'_{i_{s_i^{bw}}}=\frac{m_i^{bw}G^{bw}-(s_i^{bw})^2-2s_i^{bw}G^{bw}}{(s_i^{bw}+G^{bw})^2},$$

$$U'_{i_{s_i^{storage}}}=\frac{m_i^{storage}G^{storage}-(s_i^{storage})^2-2s_i^{storage}G^{storage}}{(s_i^{storage}+G^{storage})^2}$$

Let $U'_{i_{s_i^{cpu}}}=U'_{i_{s_i^{bw}}}=U'_{i_{s_i^{storage}}}=0$, we obtain the stagna-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2528246, IEEE Transactions on Services Computing

tion of U_i= $(\sqrt{m_i^{cpu}G^{cpu}+(m_i^{cpu})^2}-G^{cpu}$, $\sqrt{m_i^{bw}G^{bw}+(m_i^{bw})^2}-G^{bw}$, and $\sqrt{m_i^{storage}G^{storage}+(m_i^{storage})^2}-G^{storage})$

According to the sufficient extreme condition for the ternary function in [22], $U_i(s_{i0}^{cpu}, s_{i0}^{bw}, s_{i0}^{storate})$ is the maxima, and $P_0(s_{i0}^{cpu}, s_{i0}^{bw}, s_{i0}^{storate})$ is the point of maxima when A, B, C, D, E, and F satisfied

$$\begin{cases} A+D^2+1<0 \\ B+E^2+1<0 \\ C+F^2+1<0 \end{cases}.$$

Correspondingly, we obtain Equation 14.

$$A+D^2+1=\frac{-2(m_i^{cpu}G^{cpu}+G^{cpu^2})^2+1}{(m_i^{cpu}G^{cpu}+G^{cpu^2})^{\frac{3}{2}}}+1<0 \quad (19)$$

Let $H=m_i^{cpu}G^{cpu}+G^{cpu^2}$, and simplify Equation 19 to $2H^2-H^{\frac{3}{2}}-1>0$, hence we obtain $H>1$. For the same reason, utility function 12 has the maxima under the condition

$$\begin{cases} m_i^{cpu}G^{cpu}+G^{cpu^2}>1 \\ m_i^{bw}G^{bw}+G^{bw^2}>1 \\ m_i^{storage}G^{storage}+G^{storage^2}>1 \end{cases}.$$

[End of proof]

The resource bidding strategy set for the SP request was the finite positive integer set, it was greater than 1, the system had a Nash equilibrium, and the Nash equilibrium solution was in a given strategy set of the SP. The Nash equilibrium strategy is feasible.

Corollary 1. The proposed Stackelberg game system based on bid prediction is the incentive compatibility and individual rationality.

Proof: (1) Incentive compatibility: When the individual SP bids rationally, their personal profit does not suffer; at the same time, the other SPs' profits also increase, and the bid of each SP is able to achieve equilibrium s^*.

For the bid strategy of the SP, there exists that $s_i^{'}$: $U_i(s_i^*)\geq U_i(s_i^{'})$.

From the proof of Theorem 1, there exists a maximum SP 's utility function, and the extreme point of the utility function is the Nash equilibrium solution of the system. At this time, no SP has a reason to change their bidding strategy. Otherwise, it will cause loss of profit. It is clear that, when the bid of each SP has reached the state of Nash equilibrium in a Stackelberg game of imperfect information, it satisfies incentive compatible policy constraints.

(2) The individual is rational: the bidding behavior of each SP meets the individual rational constraint, and the utility is positive.

For the allocation algorithm proposed in this paper, when there is a shortage of resources some SPs may not get any. The profit for SPs that cannot obtain the resources is 0. As for SPs involved in the transaction, we discuss the possible range of Equation 12. For example, the difference between the actual valuation and the transaction price of the CPU was as follows.

$$m_i^{cpu}-s_i^{cpu^*}=m_i^{cpu}-(\sqrt{m_i^{cpu}G^{cpu}+(m_i^{cpu})^2}-G^{cpu})=(m_i^{cpu}+G^{cpu})-\sqrt{m_i^{cpu}(m_i^{cpu}+G^{cpu})}$$

For the transaction price, it is clear that $(m_i^{cpu}+G^{cpu})>\sqrt{m_i^{cpu}(m_i^{cpu}+G^{cpu})}$. That is, $m_i^{cpu}-s_i^{cpu^*}>0$ ($s_i^{cpu^*}$ presents the equilibrium). Similarly, $m_i^{bw}-s_i^{bw^*}>0$ and $m_i^{storage}-s_i^{storage^*}>0$. Equation 17 is greater than 0. All the SPs ' individual utilities are non-negative, so the Stackelberg game model of imperfect information is consistent with individual rationality.

[End of proof]

5 The Cloud Resource Allocation Model Using Hidden Markov Based on Resource Pricing

5.1 Problem Description

The cloud resource allocation model was formulated as a 4-tuple of A=<K,X,O,C>, where *K* was the set of *S* SPs. K=<k1, k2, …, ks> and X=<x1, x2, …, xs> were the price set offered by SPs. For $\forall x_i$, $x_i=<s_i^{cpu}, s_i^{bw}, s_i^{storage}>$ represented the SP i' s bid of CPU, bandwidth, and storage. O=<o1, o2, …, os> represented the combinatorial demand of all types of resources requested by the SPs. For $\forall o_i$, $o_i=<v_i^{cpu}, v_i^{bw}, v_i^{storage}>$ represented the SP i' s demand for all types of resources; C=$(c^{cpu}, c^{bw}, c^{storage})$ presented the resources of CPU, bandwidth, and storage belonging to the INs.

The economic profit of resource allocation can be expressed as an optimal model.

$$\max \sum_{i=1}^{s}\delta_i(s_i^{cpu}+s_i^{bw}+s_i^{storage})$$

$$\text{s.t.} \quad \sum_{i=1}^{s}\delta_i v_i^{cpu}\leq c^{cpu}, \sum_{i=1}^{s}\delta_i v_i^{bw}\leq c^{bw}, \sum_{i=1}^{s}\delta_i v_i^{storage}\leq c^{storage} \quad (20)$$

5.2 Description of CSAM-IISG

The resource allocation strategy in our research allowed an IN to satisfy the demands of multiple SPs simultaneously. When a SP had a trading of a set of resources, the IN allocated the requested resources to the SP all at once, rather than allocating the separate resources many times. This resulted in greatly improved system performance. During the process of resource allocation, all types of resources were pricing, and used the length of the taxonomic unit resource price vector as the greedy selection criteria. To do so, the process effectively avoided the shortcomings of the traditional allocation process of resource combining. Since the traditional allocation pro-

cess sorts SPs using the average price of all types of resources, it can cause pricing errors and form the wrong incentives [23–25].

Suppose that the requested resources are CPU, bandwidth, and storage. The resources were applied by two SPs. The resource combination pack of SP *A* was o_A=<*1, 3, 5*>, and the unit price of the resource combination was x_A=<*14, 4, 6*>. The average price of the resource combination was $\bar{x}_A = \frac{14*1+4*3+6*5}{1+3+5} = 6.22$. That of SP *B* was o_B=<*3, 1, 5*>, x_B=<*12, 2, 5*>, and $\bar{x}_B = \frac{12*3+2*1+5*5}{1+3+5} = 7$

Clearly, SP *A*'s price of each resource was higher than SP *B*'s. From the perspective of economics, SP *A* should have been more competitive than SP *B*. However, because the average price of SP *A* was lower than that of SP *B*, SP *A* was behind SP *B* in the traditional allocation. In our study, SPs were sorted using the greedy resource allocation algorithm, and we allocated resources according to the price vector lengths of unit resources.

The details of CSAM-IISG are as follows:

(1) The *s* SPs obtain their auction bids $x_i = < s_i^{cpu}, s_i^{bw}, s_i^{storage} >$ for various types of resources and their resource demand $o_i = < v_i^{cpu}, v_i^{bw}, v_i^{storage} >$ through the dynamic non-cooperative Stackelberg game pricing strategy, and submit the bids to the auction broker.

(2) The IN submits the lowest transaction price of all types of resources to the auction broker.

(3) The auction broker determines the SP participated in resource allocation according to the lowest price submitted by the IN.

(4) The auction broker calculates the resource SP prices according to the bids and demands requested by the resource provider.

$$q_i^{cpu} = s_i^{cpu} / v_i^{cpu},\ q_i^{bw} = s_i^{bw} / v_i^{bw},\ q_i^{storage} = s_i^{storage} / v_i^{storage}$$

(5) The price vector lengths $\hat{q}_i = \sqrt{(q_i^{cpu})^2 + (q_i^{bw})^2 + (q_i^{storage})^2}$ of the unit resources provided by the SP are sorted in descending order. If $\hat{q}_i$ are the same, they are sorted in descending order according to the total number $v_i^{cpu} + v_i^{bw} + v_i^{storage}$ of the requested resources.

(6) We calculate the SP's demand of various resources from the beginning of the first SP in the sorted list. If the demand is satisfied, then the resources are allocated for the SP.

(7) When the demand for the three types of resources cannot be satisfied simultaneously, we select the next SP from the sorted SP list until a class of the resource of the IN is 0 or the SP list is completely covered.

The structure of CSAM-IISG is shown in Fig. 4.

In the upper frame is the prediction based dynamic Stackelberg game pricing module among SPs. The historical resource requirement sequence $v_j^1, v_j^2, ..., v_j^n$ of any SP *j* is the observable value; while the actual bid *A* of each round is non-public information. This hypothesis is in line with the current market environment. The lower frame shows the classified resource price based cloud resource allocation module. There was a problem of lack of fairness caused by net resources sorting with the average unit price in the traditional auction model. To solve this, we priced each resource and determined the resource allocation method according to the price vector as the scoring standard basis. At the beginning of each round of resource allocation, every SP observed each competitor the resources demand. After that, each service provider used an HMM based prediction mechanism to obtain a competitor's most likely bidding strategy based on their historical resource needs. All the SPs involved in the resource auction determined their own bid price through CSAM-IISG. Algorithm 1 [28–30] is the algorithm of the multi-winner resource allocation.

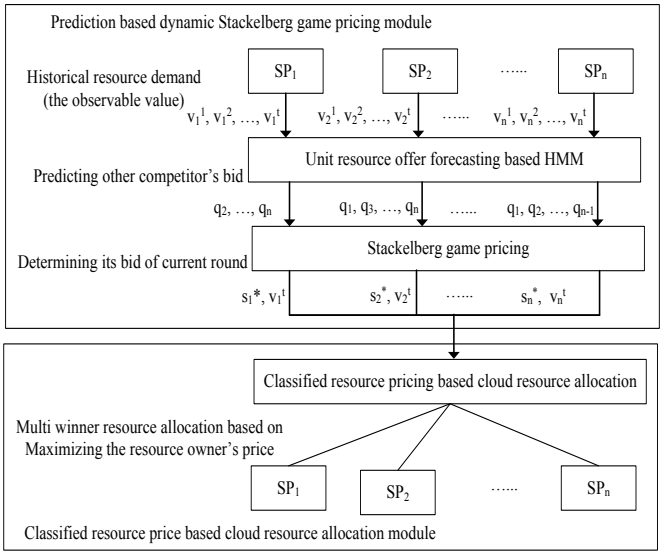


Figure 4. The structure of CSAM-IISG

Algorithm1: The algorithm of multi-winner resource allocation

```
ResourceAssignment (){
/* n indicates the number of bidders. */
/* request[i] indicates the resource demand of bidder i */
/* value[i] indicates the bid price*/
/* R indicates the total amount of resources*/
/* m indicates remaining demand */
   temp=0;
   for (i=1,2,…,n) {
     if ((request[i]+temp)≤ R)
       {profit+=value[i];
          temp+=request[i];}
    else if ((request[i]+temp)> R){
      int m=R+temp;
      double xi=(double) m/request[i];
       profit+=value[i]*xi;
       temp+= m;
       end
     }
}
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2528246, IEEE Transactions on Services Computing


```
        return profit;
    }
```

6 Performance Analysis and Evaluation of CSAM-IISG

6.1 Parameter Setting and Analysis

Suppose that the resources requested by the SP were a combination of CPU, bandwidth, and storage. The CPU price was within the range of [2, 20], the bandwidth price was within the range of [8, 40], and the storage price was within the range of [1, 10]. The unit of price was virtual resource currency (VSC). The actual value of SPs for INs is described in Equations 16 to 18:

$$m_i^{cpu}(t_i^{cpu}, v_i^{cpu}, \theta_i^{cpu}) = \eta((1-\theta_i^{cpu})\ln t_i^{cpu} + \theta_i^{cpu}\ln v_i^{cpu}) \quad (21)$$

$$m_i^{bw}(t_i^{bw}, v_i^{bw}, \theta_i^{bw}) = \mu((1-\theta_i^{bw})\ln t_i^{bw} + \theta_i^{bw}\ln v_i^{bw}) \quad (22)$$

$$m_i^{\text{storage}}(t_i^{\text{storage}}, v_i^{\text{storage}}, \theta_i^{\text{storage}}) = \omega((1-\theta_i^{\text{storage}})\ln t_i^{\text{storage}} + \theta_i^{\text{storage}}\ln v_i^{\text{storage}}) \quad (23)$$

t_i^{cpu}, t_i^{bw} and t_i^{storage} were the expected times using CPU, bandwidth, and storage by the SP; $v_i^{cpu}, v_i^{bw}, v_i^{\text{storage}}$ were the demands for CPU, bandwidth, and storage by the SP; and $\theta_i^{cpu}, \theta_i^{bw}, \theta_i^{\text{storage}}$ were the SP‘s preferences for the use time and resource demand respectively. η, μ, ω were the profits caused by the unit effects of CPU, bandwidth, and storage respectively. In the experiment, $\theta_i^{cpu} = 0.6$, $\theta_i^{bw} = 0.9$, $\theta_i^{\text{storage}} = 0.2$, $\eta = 40, \mu = 100, \omega = 20$, and 15 SPs and INs had 200 units of all types of resources.

6.2 Comparisons of the Provider’s Actual Valuation, Transaction Price, and Predicted Price

For any SP, they obtained the CPU and storage of the actual valuation, transaction price, and predicted price after 30 rounds of training using the Stackelberg game shown, as shown in Fig. 5.

In Fig. 5, the results of CSAM-IISG show that the SP‘s transaction price had the same trend as the actual valuation, and fluctuated below the actual valuation with small amplitude. According to the profit equation, it was clear that the profit of each SP was non-negative, so the CSAM-IISG model was consistent with individual rationality and reflected the actual market situation. That the predicted price obtained by the HMM fluctuated around the transaction price with small amplitude illustrated the effectiveness of the HMM.

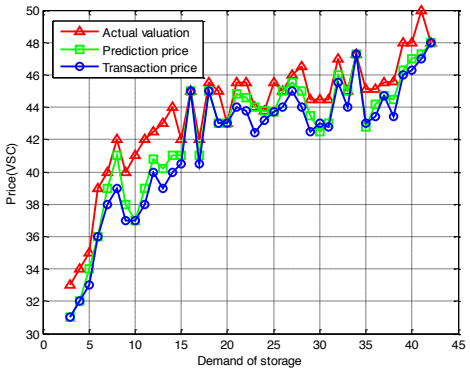


Figure 5. Comparison of an SP‘s actual valuation, transaction price, and predicted price

6.3 Profit Comparisons

The parameters of CPU, bandwidth, and storage were the same as those in Scenario 1. The resources held by the INs were set at 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100, and the number of SPs was set at 10, 30, and 50. We compared the profits of CSAM-IISG with the combinatorial traditional auction based on mean allocation. The results are shown in Fig. 6.

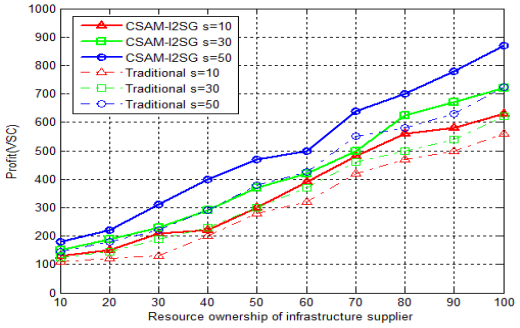


Figure 6. Profit of an IN. Note that “Traditional” represents the combinatorial traditional auction based on mean allocation

As shown in Fig. 6, the profit of CSAM-IISG was higher than that of a traditional auction when the number of SPs was the same. With an increase in SPs, the profits of both algorithms increased gradually, showing that the algorithms were consistent with the actual market rules. As the resources of INs increase, the profit of CSAM-IISG increased more significantly than that of the traditional auction. The result indicated that the proposed algorithm had a good incentive mechanism and increased the resource utility of the IN.

6.4 Comparisons of the Resource Utility of an IN

With the unit price of CPU, bandwidth, and storage the same as in Scenario 1, the number of SPs was 30, and the resource price variances were 0, 0.5, 10, 50, and 100. In the experiment, the prices of CPU and storage were kept at 6~10 and the price of bandwidth increased gradually. We experimented 20 times. Fig. 7 shows the results of the resource utility of an IN yielded by CSAM-IISG and by a combinatorial traditional auction based on mean alloca-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2528246, IEEE Transactions on Services Computing

tion.

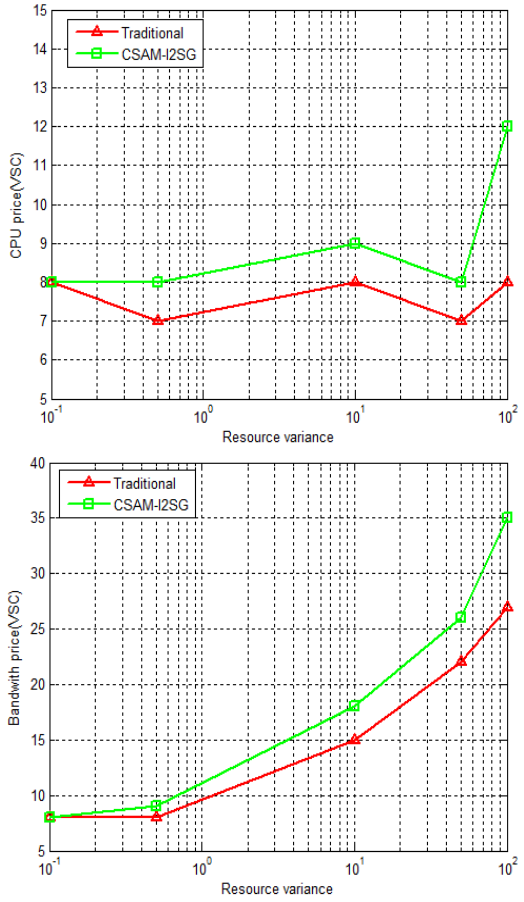


Figure 7. Comparison of resource prices sold by an IN. Note that "Traditional" represents the combinatorial traditional auction based on mean allocation

With an increase in resource price variance, the bandwidth price obtained by CSAM-IISG was clearly higher than that of the traditional algorithm, and the unit price difference increased gradually. As shown in Fig. 5, when the resources provided by the IN were limited, and the price of each resource varied greatly, the CSAM-IISG could effectively improve the resource unit prices of INs and obtain higher profits. CSAM-IISG made an SP with a higher bid more competitive, which was more consistent with the rules of market competition.

6.5 The Operation Time of Different Algorithms

Table 1 compares the computation times of CSAM-IISG with those of a traditional algorithm. The table shows that the computation time of a traditional algorithm was much greater than that of the CSAM-IISG algorithm. This was because a traditional algorithm uses only a simple crossover and mutation, while the CSAM-IISG algorithm considered the optimal characteristics of a migration, which improved the convergence rate. As a result, CSAM-IISG needs less computation time.

Table 1 Operation times of different algorithms

Task number	CSAM-IISG	Traditional
10	1.311	4.335
20	1.573	7.356

6.6 Cloud Resource Allocation System Verifications

To verify the proposed method, we adopted the cloud resource hierarchy shown in Fig. 8, and developed the cloud resources distribution monitoring model, which is supported by Huawei Co. Ltd. When a user sent a request to a virtual machine, the virtual machine could be allocated to the host if it was not satisfied. The virtual machine manager then destroyed the virtual machine and updated the host queue from the virtual machine queue, which could satisfy the virtual machine. The virtual machine scheduling module was responsible for resource scheduling. The host power management module and the data center load forecasting module were responsible for power management. The virtual machine management module was responsible for the simple scheduling of the virtual machine. The virtual machine scheduling module was responsible for scheduling the virtual machine, the operation of the improved ant colony algorithm, and the use of virtual machine migration technology to implement the scheduling results.

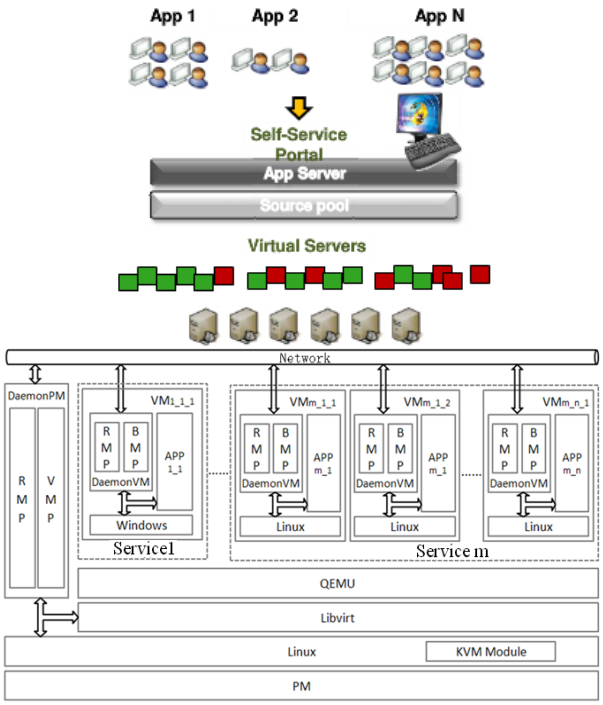


Figure 8. The cloud resources distribution monitoring model

The strategy of selecting the virtual machine to migrate involved:

- The migration minimization strategy: When the node CPU utilization ratio exceeded an upper bound, the minimum number of virtual machines was selected for migration.
- The most potential growth strategy: When the

node CPU utilization ratio exceeded an upper bound, the choice of CPU utilization was the lowest of the virtual machine migration.

- The random selection strategy: When the CPU utilization ratio exceeded an upper bound, that selected the part of the virtual machine to migrate.

When the dynamic migration ran to a certain stage, the destination power manager (PM) already had the capability of running VM3 resources. After a very short time of switching, the source PM transferred the control of VM3 to the destination PM, and VM3 continued to run on the destination PM.

Algorithm 2: Minimize migration

```
Input: hostlist, vmlist;
Output: migrationlist;
for in each host H in hostlist
    add the virtual machine running on host h to vmlist;
    sort the virtual machine in descending order according to CPU utilization in vmlist;
    save the CPU utilization of the host in hUtil;
      bestFitUtil=MAX;
      THRESH_UP = the upper bound of the host CPU utilization ratio;
      THRESH_LOW = the lower bound the host CPU utilization ratio;
end for
while hUtil>THRESH_UP
    for vmlist in each VM
      if the utilization of VM >hUtil-THRESH_UP
            t=VM -hUtil+THRESH_UP;
      if t<bestFitUtil
            bestFitUtil=t;
            bestFitVm= VM;
      else
            bestFitUtil=MAX;
            bestFitVm= VM;
      end if
            hUtil = CPU utilization;
            add bestFitVm to migrationlist;
            remove bestFitVm from vmlist;
    end for
      if hUtil<THRESH_LOW
            add the virtual machine on h to the migrationlist;
            remove the virtual machine on h from vmlist;
      end if
end while
return migrationlist
```

Figure 9. The effect of the dynamic transfer operation on the virtual machine

After the migration, VM3 ran on the destination program manager 2 (PM2), which implemented the service session without interrupting migration. CPU utilization of source program manager 1 (PM1) and destination PM2 were balanced, effectively improving overall system resource utilization.

In order to save the virtual machine, the virtual machine was integrated into a host on the virtual machine scheduling, and the virtual machine was closed down after the virtual machine migration.

The virtual machine initialization algorithm created the virtual machine to the host with the smallest power consumption. The trust driven virtual machine scheduling algorithm assigned resources to meet the needs of the trust, and the virtual machine migration algorithm was migrated to the appropriate host.

7 Conclusion and Future Work

Since cloud computing's commercial aspect makes it focus on user quality of service, the virtualization technology of cloud computing makes job and resource scheduling significantly different from those of parallel and distributed computing. Taking into account the multi-tenancy market operation mode competing in the cloud environment, this paper proposes a cloud resource allocation model based on an imperfect information Stackelberg game using an HMM. Because of the competitive relationship among SPs under multi-tenancy in the cloud environment, SPs cannot open their actual bid, so the HMM was applied to predict the SP's bid according to their historical resource demand. We propose using CSAM-IISG to determine the price, optimize the profit according to the current market situation quickly and flexibly, and achieve a Nash equilibrium. This will maximize IN and SP profits simultaneously. We designed the resource allocation model based on a classified resource price to support multiple SPs and multiple resource allocations simultaneously, which optimized IN profit. Simulation results showed that the predicted price of CSAM-IISG was close to the actual transaction price, and the transaction price was lower than the actual valuation. The proposed approach can guarantee the profits of SPs and INs. For future work, we will optimize the application

system to make it run much more efficiently and adjust the parameters according to the requirements.

ACKNOWLEDGMENTS

The authors wish to thank the anonymous reviewers for their valuable comments. This work is supported by:

- The West Virginia Higher Education Policy Commission under Grant No. FRT2W762W
- The Beilin District 2012 High-tech Plan, Xi' an, China, under Grant No. GX1504
- The Xi 'an Science & Technology project under Grant No. CXY1440(6)
- The China Postdoctoral Science Foundation under Grant No. 2013M542370
- The Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant No. 20136118120010
- The Natural Science Foundation of China under Grant No. 61272509
- The Beijing Natural Science Foundation under Grant No. 4132049
- The Shaanxi Provincial Education Department Scientific Research Program under Grant No. 2013JK1139.

REFERENCES

[1] Wu H S, Wang C J, Xie J Y, TeraPELB. An algorithm of prediction-based elastic load balancing in cloud computing [J], Journal of System Simulation, 2013, 25(8): 1751—1760

[2] Mishra M, Das A, Kulkarni P, et al. Dynamic Resource Management Using Virtual Machine Migrations [J]. IEEE Communications Magazine (S0163–6804), 2012, 50(9): 34–40. http://dx.doi.org/10.1109/MCOM.2012.6295709

[3] Zhang W, Wen Y, and Guan K, et al. Energy-optimal mobile cloud computing under stochastic wireless channel[J]. IEEE Transactions on Wireless Communications, 2013, 12(1): 4569–4581. http://dx.doi.org/10.1109/TWC.2013.072513.121842

[4] Rahman M R, Aib I, Boutaba R. Survivable Virtual Network Embedding [J]. Lecture Notes in Computer Science, 2010, 6091:40–52. http://dx.doi.org/10.1007/978-3-642-12963-6_4

[5] Chowdhury M, Rahman M R, Boutaba R. ViNEYard: Virtual network embedding algorithms with coordinated node and link mapping [J]. Networking, 2012, 20(1):206–219. http://dx.doi.org/10.1109/TNET.2011.2159308

[6] Beloglazov A, Abawajy J H, Buyya R. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing [J]. Future Generation Computer Systems (S0167-739X), 2012, 28(5): 755–768. http://dx.doi.org/10.1016/j.future.2011.04.017

[7] Yu M L, Yi Y, Rexford J, et al. Rethinking virtual network embedding: Substrate support for path splitting and migration [J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 17–29. http://dx.doi.org/10.1145/1355734.1355737

[8] Li L, Liu Y A, Liu K M, Ma X L, Yang M. Pricing in combinatorial double auction-based grid allocation model[J]. Journal of China universities of Posts and Telecommunications, 2009, 16(3)59–65. http://dx.doi.org/10.1016/S1005-8885(08)60228-9

[9] Prasad A S, Rao S. A mechanism design approach to resource procurement in cloud computing [J]. IEEE Trans. on Computers, 2014, 63(1)17–30. http://dx.doi.org/10.1109/TC.2013.106

[10] Huu T T, Tham C K. An auction-based resource allocation model for green cloud computing. 2013 IEEE International Conference on Cloud Engineering (IC2E), 25–27 March 2013: 269–278

[11] Ghazar T, Samaan N. Pricing utility-based virtual networks [J]. IEEE Trans on Network and Service Management, 2013, 10(2): 119–132. http://dx.doi.org/10.1109/TNSM.2013.043013.120304

[12] Rahman M R, Boutaba R. SVNE: Survivable virtual network embedding algorithms for network virtualization [J]. IEEE Trans on Network and Service Management, 2013, 10(2): 105–118. http://dx.doi.org/10.1109/TNSM.2013.013013.110202

[13] Zaheer F E, Xiao J, Boutaba R. Multi-provider service negotiation and contracting in network virtualization[C]. Proceedings of the IEEE Network Operations and Management Symposium (NOMS), Osaka, 2011: 471–478.

[14] Trinh T, Esaki H, Aswakul C. Quality of service using careful overbooking for optimal virtual network resource allocation[C]. Proceedings of IEEE ECTI Conference, Fort Worth, 2011: 296~299. http://dx.doi.org/10.1109/ecticon.2011.5947831

[15] Z. Zhou, M. Dong, K. Ota, R. Shi, Z. Liu, T. Sato, "Game-theoretic approach to energy-efficient resource allocation in device-to-device underlay communications". IET Communications 9(3): 375-385, 2015.

[16] I. Butun, M. Erol-Kantarci, B. Kantarci, H. Song, "Cloud-centric Multi-level Authentication as a Service for Secure Public Safety Device Networks", IEEE Communications Magazine, 54(4), 2016.

[17] M. Dong, H. Li, K. Ota, L. T. Yang, H. Zhu, "Multicloud-Based Evacuation Services for Emergency Management". IEEE Cloud Computing 1(4): 50-59, 2014.

[18] Lin W Y, Lin G Y, Wei H Y. Dynamic auction mechanism for cloud resource allocation[C]. Proceedings of IEEE/ACM 10th International Conference Cluster, Cloud and Grid Computing (CCGRID'10), Melbourne, 2010:591–592.

[19] Wang X Y，Wang X W，Huang M. Cloud resource allocation method and bidding strategy based on simultaneous upward bidding auction [J]. Journal of Northestern University, 2013, 34(4): 482–494.

[20] Chaisiri S, Lee B S, Niyato D. Optimization of resource provisioning cost in cloud computing [J]. IEEE Trans on Services Computing, 2012, 5(2): 164–177. http://dx.doi.org/10.1109/TSC.2011.7

[21] Ridder F, Bona A. Four risky issues when contracting for cloud services[R]. Research Report G00210385, Gartner, Feb. 2011.

[22] Marian M, Yong M T. Dynamic resource pricing on federated clouds[C]. Proceedings of IEEE/ACM 10th Int' l Conference Cluster, Cloud and Grid Computing (CCGRID ' 10), Melbourne, 2010, 513–517.

[23] Altous, H, Barhumi I. Resource allocation for AF cooperative communications using Stackelberg game[C], 6th International Conference on Digital Object Identifier, Dec. 2012: 1 - 6.

[24] Liu S L, Wang M X. Sealed-bid auctions based on Cobb-Douglas utility function [J].Economics Letters, 2010, 107(1):1–3. http://dx.doi.org/10.1016/j.econlet.2009.05.019

[25] Cao Q, Zhao H V, Jing Y. Power allocation and pricing in multiuser relay networks using Stackelberg and bargaining games[J]. IEEE Trans. Vehicle Technology, 2012, 6(7): 3177–3190 http://dx.doi.org/10.1109/TVT.2012.2204076

[26] Li M C, Xu L, Sun W F. Grid resource allocation model based on incomplete information game[J]. Journal of Software, 2012, 23(2): 428–438. http://dx.doi.org/10.3724/SP.J.1001.2012.03972

[27] Liu D, Chen Y, Vinel A. Stackelberg game based cooperative user relay assisted load balancing in cellular networks [J], IEEE Communication Letter, 17(2): 424–427, 2013

http://dx.doi.org/10.1109/LCOMM.2012.122012.122377

[28] Lin W Y, Lin G Y, Wei H Y. Dynamic auction mechanism for cloud resource allocation[C]. Proceedings of IEEE/ACM 10th International Conference Cluster, Cloud and Grid Computing (CCGRID ' 10), Melbourne, 2010:591–592

[29] Chaisiri S, Lee B S, Niyato D. Optimization of resource provisioning cost in cloud computing [J]. IEEE Trans on Services Computing, 2012, 5(2): 164 – 177

[30] Niyato D, Wang P, Hossain E et al. Game theoretic modeling of cooperation among service providers in mobile cloud computing environments[C]. Proceedings of the IEEE Wireless Communications and Networking Conference: Services, Applications, and Business, 2012: 3128–3133.

Wei Wei received his Ph.D. and M.S. degrees from Xian Jiaotong University in 2011 and 2005, respectively. Currently he is an assistant professor at Xian University of Technology. His research interests include wireless networks and wireless sensor networks applications, mobile computing, distributed computing, and pervasive computing.

Xunli Fan received his M.S. and Ph.D. degrees from Northwestern Polytechnical University in 1996 and 2000, respectively. Currently he is an associate professor at Northwest University. His current research interests focus on network QoS and wireless sensor networks.

Houbing Song (M'12-SM'14) received a Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, in August 2012.

In August 2012, he joined the Department of Electrical and Computer Engineering, West Virginia University, Montgomery, WV, where he is currently an assistant professor and the founding director of both the West Virginia Center of Excellence for Cyber-Physical Systems (WVCECPS), sponsored by the West Virginia Higher Education Policy Commission, and the Security and Optimization for Networked Globe Laboratory (SONG Lab, www.SONGLab.us). His research interests lie in the areas of cyber-physical systems, the Internet of things, cloud computing, big data, connected vehicles, wireless communications and networking, and optical communications and networking. Dr. Song's research has been supported by the West Virginia Higher Education Policy Commission.

Dr. Song is a senior member of IEEE and a member of ACM. Dr. Song is an associate editor for several international journals, including IEEE Access, KSII Transactions on Internet and Information Systems, and SpringerPlus, and a guest editor of several special issues. Dr. Song was the general chair of four international workshops, including the first IEEE International Workshop on Security and Privacy for the Internet of Things and Cyber-Physical Systems (IOT/CPS-Security), held in London, UK, and the first/second/third IEEE ICCC International Workshop on the Internet of Things (IOT 2013/2014/2015), held in Xi'an/Shanghai/Shenzhen, China. Dr. Song also served as the technical program committee chair of the fourth IEEE International Workshop on Cloud Computing Systems, Networks, and Applications (CCSNA), held in San Diego, USA. Dr. Song has served on the technical program committee for numerous international conferences, including ICC, GLOBECOM, INFOCOM, and WCNC. Dr. Song has published more than 80 academic papers in peer-reviewed international journals and conferences.

Xiumei Fan is a professor in the School of Automation and Information Engineering, Xi'an University of Technology. She received a B.S. degree in electron information engineering from Tianjin University in 1989 and a Ph.D. degree in communication and information systems from Northern Jiaotong University in 2001. She has finished some projects of NSFC and The National HighTechnology Research and Development Program of China in Delay Tolerant Networks, Vehicular Networks, and Mobile Internet. She is now leading two advanced wireless network projects under NSFC and SRFDP. Her current research interests focus on wireless broadband networks, VANET, DTN, and the mobile Internet. She has published more than 60 papers. She is also a distinguished professor of the one hundred plan in Xi'an University of Technology.

Jiachen Yang received M.S. and Ph.D. degrees in Communication and Information Engineering from the Tianjin University, Tianjin, China, in 2005 and 2009, respectively. He is an associate professor at Tianjin University. In 2014, he was a visiting scholar in the Department of Computer Science, School of Science, at Loughborough University, UK. His research interests include intelligent transportation systems, machine learning, pattern recognition, stereo image displaying, and quality evaluation.