Performance-Aware Cloud Resource Allocation via Fitness-Enabled Auction

Hongbing Wang, *Member, IEEE*, Zuling Kang, and Lei Wang

Abstract—Cloud computing is a new computing paradigm which features renting the computation devices instead of buying them. In a typical cloud computing environment, there will always be different kinds of cloud resources and a number of cloud services making use of cloud resources to run on. As we can see, these cloud services usually have different performance traits. Some may be I/O-intensive, like those data querying services, while others might demand more CPU cycles, like 3D image processing services. Meanwhile, cloud resources also have different kinds of capabilities such as data processing, I/O throughput, 3D image rendering, etc. A simple fact is that allocating a suitable resource will greatly improve the performance of the cloud service, and make the cloud resource itself more efficient as well. In this paper, a new cloud resource allocating algorithm via fitness-enabled auction is proposed to guarantee the fitness of performance traits between cloud resources (sellers) and cloud services (buyers). We study the allocating algorithm in terms of economic efficiency and system performance, and experiments show that the allocation is far more efficient in comparison with the continuous double auction in which the idea of fitness is not introduced.

Index Terms—Performance-aware, cloud resource allocation, fitness-enabled auction, cloud computing, auction theory

1 Introduction

Cloud computing [1], [2], [3] is becoming a new computing paradigm, and a dominant IT news topic over the past years. By renting instead of buying computation power, this new computing paradigm enables its users to freely increase or decrease their computation power, and pay according to the usage of the quality and quantity of computation. Currently, the main technical underpinnings of cloud computing include virtualization, service-oriented architecture (SOA), distributed & parallel computing, resource management, etc. Although none of these technologies are new to us, they do exhibit new challenges under the cloud environments. And resource allocation is one of them.

New challenges of resource allocation in cloud computing lie in the fact that the cloud platform is built for commercial uses, which demands that all the resources in the cloud platform be paid for uses. In such cases, there are at least two new requirements to be considered: one is the trade-offs between prices and profits for cloud resource providers (resource agents), or prices and the required level of QoS for resource renters (service agents); the other is fitness between resources and services. Moreover, these new requirements are not totally independent of each other. Allocating a suitable resource to a certain service will improve both the profits for the provider and the QoS level for the renter, while providing a good price for both of them. As for that, the two mutually related requirements constitute the problem lies ahead for the resource allocating problem in cloud computing.

The idea of our approach originates from the continuous double auction (CDA) [4], [5], which is a kind of market-based resource management method, and has received much attention in the recent years [6], [7], [8], [9], [10]. It enables the regulation of supply and demand for resources, and motivates the resource providers and renters to trade-off between prices, profits, and the required level of QoS [7]. Market-based methods have proven effective in dealing with many resource allocation problems for both grid [6], [7], [10] and cloud computing [8], [9], [11], [12]. Nevertheless, none of the existing market-based resource allocation methodologies have considered the problem of performance fitness. In the situation of a practical application environment, there usually exist different kinds of resource renters (e.g., I/O-intensive services or CPU-intensive services) require for different resources of performance traits.

In this paper, we present a new resource allocating algorithm named Cloud Resource Allocating Algorithm via Fitness-enabled Auction (CRAA/FA), to deal with these new challenges in cloud resource allocation. Like any other resource allocation methods that employ auction, CRAA/FA models the cloud platform as the resource market, while the cloud resource providers (cloud resource agents) and renters (cloud service agents) as sellers and buyers, respectively. During the allocation period, buyers submit bids and sellers submit requests at any time during the trading period. At any time when there is a bid and request that matches or is compatible with a price, a trade is executed immediately [6].

Since the fitness-enabled auctionoriginates from the idea of CDA [4], [5], FA follows the basic principles of CDA.

• *The authors are with the School of Computer Science and Engineering and Key Laboratory of Computer Network and Information Integration, Southeast University, SIPAILOU 2, NanJing 210096, China, and China Mobile Group Zhejiang Co., Ltd, Hangzhou 310016, China. E-mail: {hbw, leiwang}@seu.edu.cn, zelkey@msn.com.*

Manuscript received 19 Aug. 2014; revised 21 Feb. 2015; accepted 18 Apr. 2015. Date of publication 23 Apr. 2015; date of current version 16 Mar. 2016. Recommended for acceptance by M. Steinder.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2015.2426188

However, it introduces a new measure, *fitness*, and a new asking/bidding strategy, *the dynamic asking/bidding strategy*, into the bargaining model, so as to improve the auction efficiency in our intended cloud environment. Moreover, to make the fitness measure work efficiently during auction, CRAA/FA modifies the bargaining process as well as the expression to calculate the final price. In sum, resource providers and renters in FA can autonomously arrive at trade-offs between prices, profits, and the required level of QoS, and the fitness measure works potentially during the whole auctioning process, finding the suitable resources or services, calculating the equivalent prices, and getting the final price value. Experiments show that our FA-based auction method is efficient not only for a single provider or renter, but also for the whole cloud platform or the whole cloud resource market.

In respect with the fact that the introduction of fitness as well as the dynamic asking/bidding strategy do improve the dealing rates of the cloud resource market, we believe that the introduction of fitness will be the most important contribution of this paper. As for the resource allocation, it autonomously dispatches cloud services to their *suitable* resources, improving the overall efficiency of the whole cloud platform, the overall efficiency and profit of cloud resource providers, and the overall performance of cloud services. In this way, FA provides a better trade-off for both providers and renters. Moreover, since the fitness problem also exists in some other resource allocation markets, the approach introduced in this paper can also be applied in dealing with the auctioning problem of these markets. Thus, our work also has the universal significance in studying auction models.

1.1 Motivations to Use Auction Method

Generally speaking, supply and demand, i.e. the market mechanism, can be used to bring an efficient resource distribution between cloud services (buyers) and resources (sellers) with different QoS traits and service levels. By employing the supply and demand mechanism with the auction method, we are capable of using the *invisible hand* of the market to distribute the cloud resources to the cloud services with different resource requirements, efficiently, fairly, and without a centralized resource allocator. In summary, we can achieve three resource distributing goals via the auction method discussed in this paper: 1) a dynamic load balancing amongst cloud areas, i.e. the *dynamic balance* goal, 2) the priority of selecting more suitable resources for cloud services, i.e. the *fitness* goal, and 3) the differentiation among services with different service-level agreement, i.e. the *differentiation* goal.

On one hand, when a certain resource area A_i finds more demands (buyers) coming, it will increase its asking prices so as to gain more profits. Since the cloud services with higher levels always have more budgets, they in this way receive the priority to get the tight resources which are in great demand in the market. By this means, the *differentiation* goal is satisfied. Moreover, the increment of A_i's asking price makes other resource areas easier to make deals. In other words, it actually distributes services to resources with light load, so achieves the *dynamic balance* goal.

On the other hand, resources in different areas are often suitable with different services. For instance, I/O-intensive services should be deployed in the resources with high I/O throughput, while the image processing services are suitable with the ones installed with special image accelerators like GPUs. Informally, if the fitness value between a cloud service CS_j and a resource A_i is higher, A_i is able to provide the very computing power that CS_j needs. In such circumstance, the resource consumption of running CS_j on A_i is usually lower than the average level. Based on the discussion above, we introduce a new *equivalent price* mechanism into our auctioning process. The formal definition and further discussion of the equivalent price will be given in the sections later on. Briefly speaking, for two areas A_i and $A_{i'}$ where A_i is more suitable with CS_j than $A_{i'}$, if A_i and $A_{i'}$ ask the same price during the auctioning process, CS_j will receive a lower equivalent price from A_i, so that A_i will be easier to execute the trade with CS_j. From the viewpoint of the whole market, that the cloud services are motivated to be distributed into the more suitable resources satisfies the *fitness* goal.

1.2 Reifying of Application for Cloud Industry

In recent years, there appear several scheduling and resource allocating frameworks for the distributed computing environment, which are to simplify the complexity of running applications on a shared pool of servers. Examples of these frameworks include Hadoop YARN [13] and Apache MESOS [14], both of which are PaaS-like frameworks, and provide a plug-in mode for the resource allocators. In this way, it is much easier for users to implement their resource allocators without having to code from scratch.

Taking YARN as an example. To make our own allocation work, all we have to do are to implement the ResourceScheduler interface, and set the allocator's class name in the yarn.resourcemanager.scheduler.class parameter in yarn-site.xml. YARN and MESOS have also included some widely used resource allocators, including FIFO, the Capacity Scheduler by Yahoo! and the Fair Scheduler by Facebook. Following this idea, we can also integrate our cloud resource allocating algorithm to these existing cloud platforms, and use it in the real cloud environments.

Of course, to implement our fitness-enabled auction algorithm in YARN, details must be carefully concerned according to the programming model of YARN. First, there must be a software constructed that acts as the auctioneer of the whole market (the whole Hadoop cluster in this case), which can either be a YARN service, or a fully independent YARN application. Second, our scheduler must implement the ResourceScheduler interface by providing our own handle() method from ResourceScheduler, and interact with the rest of YARN component by messaging since YARN is an event-driven framework. Third, as YARN is now unable to hang its running application. However, it is a must for our CRAA/FA algorithm, if a cloud service (YARN application) could not acquire any resource due to its failure in an auction. So we have to hack into the source code of YARN to create a new application state, HUNG, in YARN, and provide a mechanism to hang the running application.

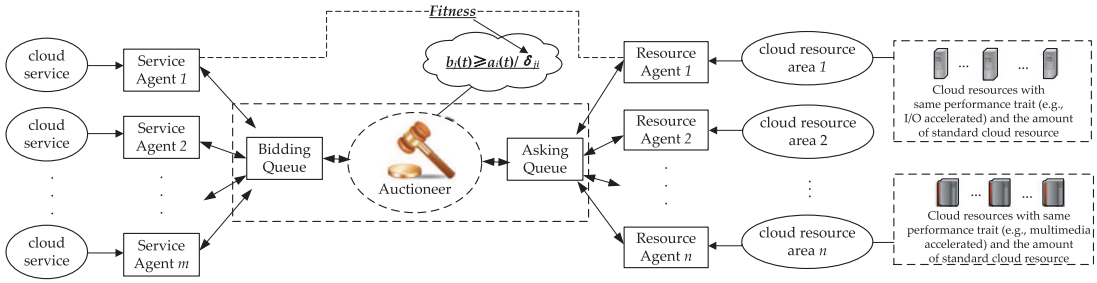


Fig. 1. Auctioning model of the FA-based cloud resource market.

This paper was first presented in a shortened form as a conference paper in [15]. In this paper, we significantly extend the CRAA/FA algorithm and present a comprehensive framework for performance-aware cloud resource allocation. The rest of this paper is organized as follows: Section 2 states the background knowledge. In Section 3, we discuss the details of the CRAA/FA algorithm. Experimental evaluations are presented in Section 4. Section 5 addresses the related works. We conclude by identifying some important future directions in Section 6.

2 Preliminaries

2.1 Auction Theory

Auctions are used for products that have no standard values and the prices are affected by supply and demand at a specific time. It requires little global price information, is decentralized, and easy to implement in distributed settings like grid computing and cloud computing [6]. Based on interactions between consumers and providers, auctions can be classified into four basic types: the ascending auction (English auction), the descending auction (Dutch auction), the first-price and second-price sealed auction, and the double auction [4], [5].

A double auction is a process of buying and selling coessential goods when potential buyers submit their bids and potential sellers simultaneously submit their asks to an auctioneer, and then an auctioneer chooses some price p that clears the market: all other sellers who asked more than p, and all other buyers who bid less than p. Generally speaking, three most popular double auctions are [6]: Preston-McAfee Double Auction Protocol (PMDA), Threshold Price Double Auction Protocol (TPDA), and Continuous Double Auction Protocol. Grosu and Das [16] showed that CDA is better from both the resource's and the user's perspective providing high resource utilization.

A double auction can be analyzed as a game. Players are buyers and sellers. They have some valuations of a good that is traded in an auction. Their strategies are bids for buyers and ask prices for sellers (that depend on the valuations of buyers and sellers). Payoffs depend on the price of the transaction and the valuation of a player.

2.2 System Model and Cloud Resource Market

The cloud resource auctioning model is shown in Fig. 1. Like traditional CDA models, our FA is a many-to-many (buyers and sellers) auctioning strategy. In the cloud resource market, the resource agents (resource providers) act as sellers, the service agents (resource renters) as buyers, and the resource allocation service as the auctioneer. Commodities exchanged in the market are the cloud resources. Specially, the cloud resources stated in this paper is defined as follows.

Definition 1 (Cloud resource). *A cloud resource is defined as a rentable entity holding specific performance trait, and can provide an execution environment for a cloud service to run on.*

In cloud industry, such as a physical server node or a logical VM instance (e.g., realized by Docker) all can be regarded as cloud resources.

In the FA-based cloud resource market, resource agents and service agents submit their asks and bids to asking queue and bidding queue, respectively. At any time, an asking or bidding price is anonymously selected and submitted to the auctioneer. The trade is executed immediately when the largest bidding price is not lower than the smallest asking price (c.f. Section 3.1). During the auction, the resource agents and service agents will stick to the pricing strategies which were set in advance. The asks and bids are the system behavior of CRAA/FA algorithm, it cannot be influenced by any outside interventions. The auctioneer does nothing but as a mediator of the market place.

The cloud resource can be quantified by unit of standard cloud resource (UCR), whose value is defined by the amount of time (say t seconds) consumed when running a *standard* cloud service over a *standard* cloud node. Informally, the *standard* cloud node is usually a *common* server designated by the cloud platform administrators. It is so *common* that there usually does not install any accelerator device like SSD-based I/O accelerator or 3D graphics accelerator in it. Similarly, the *standard* cloud service is a *common* service that is neither CPU-sensitive nor I/O-sensitive, and it also has no extra resource requirement like a huge memory requirement.

Employing the definition of the *standard* cloud service and the *standard* cloud node, we can further define that if it takes the *standard* machine nt seconds to run a cloud service CS_j, then CS_j consumes n UCRs. Furthermore, if it takes mt seconds when the same service CS_j is running on cloud resource CN_i, this means there are n/m UCRs existed within the t second running period of CN_i, and the running speed of CN_i is n/mt UCRs per second. By this means, it'll be easy to calculate the amount of standard cloud resource a cloud service consumes for every execution, and the speed of every specific type of cloud resource.

To reduce the number of resource agents, we group the cloud resources into areas. Resources in one area hold the same performance trait and UCR amount, and share the

same cost and asking price. The resources with different performance trait (e.g., I/O accelerated, multimedia accelerated, etc.) and equipment (e.g., VM instance with two threads or four threads) will be classified into categories and announced by platform administrator. Different categories of cloud resources will be put on the market by resource providers. The number of areas is determined by the number of resource categories. For a particular application, the area number is relatively fixed. Resources in an area share a same fitness vector (c.f. Section 2.3), and there is one specific resource agent for each area. Thus, sellers can be described as a set of areas, while an area V_i can further be described by the number of resource threads $|A_i|$, the speed for every resource of the area V_i, and its fitness vector $\delta_i\vec{}$.

Cloud services need cloud resources to run and can be seen as the consumers (or buyers) of the market. Each cloud service has one specific service agent. Cloud service CS_j can be measured and described by the amount of standard cloud resource D_j it consumes for one execution over the *standard* machine, and its average request rate v_j (UCRs per second). During auctioning, these two arguments can be used to calculate the amount of resource a cloud service needs to achieve its QoS level.

There are different kinds of cloud services in the real environment. In our experimental environment, we identify three of them, Type-I, Type-II, and Type-III. Type-I services feature short running time of a single service request, however there might be a huge number of service requests every second. The QoS parameters of Type-I services include the required response time T_j^{RST}, and the ratio of service requests η that are permitted to exceed the required response time. Type-II and Type-III services both feature long running time of a single service request, however they are not requested that often, e.g. just several times or tens of times a day. Every request to a Type-II service has a deadline, and users are satisfied if their requests are completed before the deadline. On the contrary, requests to Type-III services do not go with a deadline, and users of Type-III services demand their requests be completed as fast as possible. Respectively, the Type-I, Type-II, and Type-III service will typically be the OLTP service, the OLAP service with an explicit defined deadline, and the OLAP service which is required to complete as fast as possible in a practical scenario.

The differentiation of service level among different cloud services is achieved by the budget it receives for every resource renting period or every request to the cloud service. A resource renting period is a time length set by the administrator of the cloud platform, by minutes or tens of minutes. The budget of a cloud service reflects the maximum price it is able to pay for a renting period of cloud resources.

Comparing with one-side auction, CDA-based market mechanism is believed to improve the market fairness [6]. In our CRAA/FA model, both buyers and sellers can submit their bidding/asking prices freely. Bidders with high bidding prices will have higher priority to use the resources. Resources with low asking prices will more easily to be sold. The dealing takes all the bidding and asking prices among the whole market into account. Profits are the only incentives to buyers and sellers in the cloud resource market. And auctioneer is doing nothing but making the dealing judgments. It seems no one in market really cares about how to satisfy the different levels of QoS requirements of the cloud services, whether a service will find a suitable resource to run, or the profit of the whole cloud resource market. However, the invisible hand does work in background, keeping the fitness and profit, and improving the fairness of the market.

Note that the system architecture for FA-based resource allocation market presented in this paper is centralized. Although it is not a complex computing task for processing the trades, when dealing with a diverse large-scale market place, a large number of client-side concurrent connections will lead to frequent process switching for the auctioneer node. In such case, the auctioneer node will be in face of bottleneck problem. Nevertheless, we can solve the bottleneck problem via deploying distributed operating environment for auctioneer. Recently, some researchers have also investigated distributed economic-based resource allocation approaches, e.g., catallaxy-based market mechanism [17], [18]. A catallaxy-based cloud resource market depends on some catallactic (or wholesalers) agents to trade on selling nodes and buying nodes. However, there still have no role acts as wholesalers in current cloud resource market. Therefore this trading mode is not reality for current cloud resource market. Perhaps this is a new trading mode to be developed in the future, and our approach can be readily extended for this trading mode.

2.3 Fitness and Equivalent Price

Fitness is the most important contribution of this paper. It is an argument to measure how an area and a cloud service are suitable with each other. As we have addressed in the previous sections, a cloud service used to do data querying are surely to run faster in a cloud resource with an I/O accelerator installed than in a cloud resource with a mere graphics accelerator. But this argument can only be usable in auctioning when it is able to be properly measured.

Definition 2 (Fitness). *Suppose a cloud service CS_j consumes D_j units of* **standard** *cloud resource for one execution, however, when it is running on a resource in Area A_i, it just consumes D_{ji} units of A_i's cloud resource, then the fitness δ_{ji} between A_i and CS_j as:*

$$\delta_{ji} = D_j / D_{ji}.$$

As we have stated in the previous subsection, the *standard* machine does not contain any accelerating modules, the fitness value δ_{ji} is usually greater than or equal to 1. For a cloud platform deployed with m cloud services, the fitness vector $\vec{\delta_i}$ is defined as $(\delta_{1i}, \delta_{2i}, \ldots, \delta_{mi})$.

A cloud service will always prefer a resource area with a higher fitness value, because it is able to guarantee it's QoS-level requirement under such resource. Suppose CS_j needs x units of standard cloud resource, it only needs x/δ_{ji} units of A_i. If we further suppose the standard resource and the A_i resource share the same price, then CS_j is able to pay less equivalent price when renting A_i. A_i will obtain more profits for selling to CS_j.

Definition 3 (Equivalent price). *Suppose at a certain time point t during the auctioning process, A_i asks for $a_i(t)$, CS_j bids for $b_j(t)$, and the fitness between A_i and CS_j is δ_{ji}. Then A_i's equivalent price with CS_j is $a_i(t)/\delta_{ji}$.*

Equivalent price is directly related to fitness. Since CS_j only needs $1/\delta_{ji}$ in amount of standard cloud resource when choosing A_i, it's easy to image that for A_i's resource, its equivalent asking price is $a_i(t)/\delta_{ji}$ for CS_j. Based upon that definition, it is easy to see that the higher the fitness value δ_{ji} is, the more the area A_i and cloud service CS_j are suitable to each other. During the auctioning process, the fitness value would not only help resources/services to choose more suitable services/resources so that the providers, renters, and the whole resource market would gain higher economical benefits via auction, but would guarantee the specific performance-aspect requirement of each cloud service.

3 Auctioning in the Cloud Resource Market

3.1 Auctioning Process of the CRAA/FA Algorithm

The basic idea of the auctioning process in CRAA/FA is similar with other double auctions. However, with the introduction of fitness and equivalent price, its detail is totally different. Let there exit m cloud services in the market, each ask of area A_i will become m asks with different fitness values. Consequently, the auctioning process becomes more complex as we bring the fitness issue into double auction. The auctioneer will have to deal with the fitness of resources between buyers and sellers.

To reduce the computational complexity of auctioneer node by fitness-enabled auction, we only select TOP-N bidding prices and Top-M fitted cloud services for sellers, and Lowest-N asking prices and Top-M fitted cloud resource areas for buyers, and form the Active Bargaining Set, among whom the ultimate transaction would most possibly take place. Only the Active Bargaining Set would be considered by auctioneer for dealing. The Active Bargaining Set is formally defined as follows.

Definition 4 (Active bargaining set). *Denote $Q_{CS}^N(t)$ as a cloud service set with N cloud services whose agents submit higher bids than any other services not in $Q_{CS}^N(t)$ at time t, and $Q_A^N(t)$ as an area set whose agents submit lower asks than any other areas not in $Q_A^N(t)$ at time t. We further suppose $Q_j^{CS}(M)$ as an area set whose element (a cloud node) has a higher fitness value with CS_j than any other areas not in $Q_j^{CS}(M)$, and $Q_i^A(M)$ as a cloud service set whose element (a cloud service) has a higher fitness value with A_i than any other cloud services not in $Q_i^A(M)$. Thus the active bargaining set for A_i at time t is $Q_{CS}^N(t) \cup Q_i^A(M)$, and for CS_j is $Q_A^N(t) \cup Q_j^{CS}(M)$.*

The active bargaining set can be calculated by inspecting the fitness value, as well as the recent asks and bids of the cloud resource areas and services. Taking the active bargaining set, we present the auctioning process as the following.

1) Let t be a period during the auction, cloud service agent for CS_j and area agent for A_i are able to submit their bid $b_j(t)$ and ask $a_i(t)$ during t. In one hand, areas and services will update their asks and bids at a specific time (i.e., the beginning of t, $t+1$, etc.); in the other hand, the updates will also take place when the amount of resource or the number of available services (the available service means this service has not got its resource, so it is still available for other resources) changes.
2) During the whole auctioning process, the auctioneer works with two price queues, i.e. the bidding queue for cloud services and the asking queue for areas. To initialize the queues, each service and area agent submit their bidding and asking prices to the queues. Any a bidding or asking price will be sequently selected and submitted to the auctioneer. When receiving an ask $a_i(t)$ for A_i, it chooses a cloud service CS_j from the active bargaining set for A_i so that

$$CS_j = \arg\max_{CS_k} \left\{ a_i(t)/\delta_{ki} | CS_k \in Q_{CS}^N(t) \cup Q_i^A(M) \right\}.$$

   Then, if the auctioneer finds that the dealing condition $b_j(t) \geq a_i(t)/\delta_{ji}$ holds, a trade between A_i and CS_j is executed immediately; otherwise, another bidding or asking price would be selected for auction.
3) Similarly, when the auctioneer receives a bid $b_j(t)$ for CS_j, it chooses an area A_i from the active bargaining set for CS_j so that

$$A_i = \arg\min_{A_k} \{ a_k(t)/\delta_{jk} | Q_A^N(t) \cup Q_j^{CS}(M) \}.$$

   Then, if the auctioneer finds that the dealing condition $b_j(t) \geq a_i(t)/\delta_{ji}$ holds, a trade between A_i and CS_j is executed immediately; otherwise, select the next bidding or asking price for auction.
4) If a trade is executed in (2) or (3) and there are still remaining cloud resources in A_i, the area agent for A_i will update its asking price and resubmit it to the price queues.

3.2 Strategies to Determine Ask and Bid Values

We now discuss the strategies to determine the ask and bid values in this section. Like many of other works [6], [11], [12], these strategies are modeled on the idea of negotiation decision functions as proposed by Faratin et al. [19]. This model identifies multiple key constituents (or issues) that drive an agent's negotiation behavior and defines separate tactics to deal with each of them. The agent's overall behavior is then the amalgamation of these different tactics, weighted by their relative importance to the user [20].

The commonly used tactics usually include the remaining time, the remaining auctions and the desire for a bargain [6], [19], [20]. In this paper, we mainly identify two tactics for cloud services' agents, i.e. the remaining time and the potential areas; while for cloud resource's agents, we identify three tactics, i.e. the remaining time, the potential services, and the vacant resources. The choice of these significant tactics for the buyers and sellers is based on the following considerations: 1) these tactics are widely used in solving similar problems in [6], [20]. 2) There are many available tactics for CDA. But which tactics(s) should be selected that all depends on the specific circumstances. 3) When more tactics are taken into account, the

effectiveness of resource allocation and the efficiency of market profits during the auction process will be enhanced. It will however make the auction process be more complex. By a trade-off decision, only these critical tactics are used for our CRAA/FA algorithm.

The remaining time tactics for both services and resource areas are identical, so we now take the resource areas for further discuss. Since cloud resources are often preallocated, we can use the kth renting period to auction for the resources of the $(k+1)$th renting period. Suppose the length of a renting period is τ (which is actually the total bargaining period), $rp(t)$ represents the start time of a bargaining period which satisfies

$$rp(t) \le t < rp(t) + \tau,$$

then the remaining time at t will be

$$\gamma^{time}(t) = (rp(t) + \tau) - t,$$

so that

$$0 \le \gamma^{time}(t) \le \tau.$$

Denote the area A_i's maximum and minimum asking prices for every cloud resource is $a_i^{\max}$ and $a_i^{\min}$, respectively. At a certain time t, A_i's asking price is surely between $a_i^{\max}$ and $a_i^{\min}$, and decreases as it approaches the deadline of the bargain period. So A_i's asking price at time t can be computed via the following expression:

$$a_i^{rt}(t) = a_i^{\min} + \kappa^{rt}(\gamma^{time}(t))\left(a_i^{\max} - a_i^{\min}\right)$$

in which $\kappa^{rt}(\cdot)$ is a function of the remaining time. It can dynamically adjust the asking price with the varying of remaining time. It's value decreases when the remaining time drops, and it would be further explained soon. And at the beginning of the bargaining period, i.e.

$$\gamma^{time}(t) = \tau,$$

the asking price will be the highest, i.e.

$$a_i^{rt}(\tau) = a_i^{\max}$$

so that

$$\kappa^{rt}(\tau) = 1.$$

On the contrary, when it approaches the deadline of auctioning, i.e. the remaining time is almost 0

$$\gamma^{time}(t) \to 0.$$

there will be

$$a_i^{rt}(0) \to a_i^{\min}$$

. so that

$$\kappa^{rt}(0) = 0.$$

Using the expression above, we can actually create infinite number of asking functions if only $\kappa^{rt}(\tau) = 1$ and $\kappa^{rt}(0) = 0$ hold. Faratin et al. [19] and Anthony and Jennings [20] identify two of most commonly used expressions of $\kappa^{rt}(x)$ in their works, the polynomial function

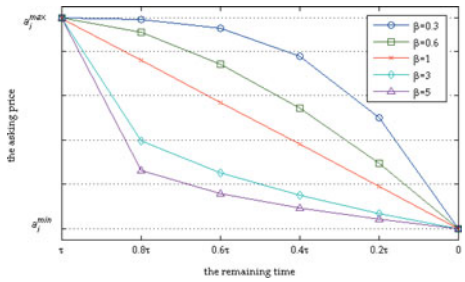


Fig. 2. Asking parameter versus asking price.

$$\kappa^{rt}(x) = \kappa^0 + (1-\kappa^0)\left(\frac{x}{x_{\max}}\right)^{\frac{1}{\beta}}$$

and the exponential function

$$\kappa^{rt}(x) = exp\left\{\left(1 - \frac{x}{x_{\max}}\right)^{\beta} \cdot \ln \kappa^0\right\}.$$

In the expressions above, $\beta \in R^+$ stands for the curvature of the asking function in its function graph, and κ^0 is any real number (constant) between 0 and 1. Specifically, when

$$\beta \to 0^+$$

there will be

$$\lim_{\beta \to 0^+} \kappa^0 + (1-\kappa^0)\left(\frac{x}{x_{\max}}\right)^{\frac{1}{\beta}} = \kappa^0$$

$$\lim_{\beta \to 0^+} exp\left\{\left(1 - \frac{x}{x_{\max}}\right)^{\beta} \cdot \ln \kappa^0\right\} = \kappa^0$$

and when

$$\beta \to +\infty$$

there will be

$$\lim_{\beta \to +\infty} \kappa^0 + (1-\kappa^0)\left(\frac{x}{x_{\max}}\right)^{\frac{1}{\beta}} = 1$$

$$\lim_{\beta \to +\infty} exp\left\{\left(1 - \frac{x}{x_{\max}}\right)^{\beta} \cdot \ln \kappa^0\right\} = 1.$$

Moreover, it is easier to conclude that, as far as the polynomial and exponential functions are concerned, when β is larger, the polynomial function concedes faster than the exponential one; and when β is smaller, it is just the opposite [19].

Taking the polynomial function, A_i's asking function on the remaining time tactic can be defined as

$$a_i^{rt}(t) = a_i^{\max} - \left(a_i^{\max} - a_i^{\min}\right)\left((\tau - \gamma^{time}(t))/\tau\right)^{\frac{1}{\beta}}$$

in which β is called the asking parameter that controls both the directions and degrees (curvatures) of the asking curve. Fig. 2 shows the asking curves for $\beta = 0.3$, 0.6, 1, 3 and 5. And as we learnt from the figure, when $\beta > 1$, the curve appears concave and its curvature increases as the value of β increases. This means the asking strategy of the resource agent is more conservative when $\beta > 1$, i.e. it tends to keep its asking price closer to $a_i^{\min}$ even when there is still much

remaining time left. On the contrary, when $0 < \beta < 1$, the asking curve appears convex and curvature increases as β decreases. This means the asking strategy of the resource agent is more aggressive when $0 < \beta < 1$, i.e. it tends to keep it asking price as high as possible unless there is really little time left.

Similarly, we can also get CS_j's bidding function on the remaining time tactic as

$$b_j^{rt}(t) = b_j^{\min} + \left(b_j^{\max} - b_j^{\min}\right)\left((\tau - \gamma^{time}(t))/\tau\right)^{\frac{1}{\beta}}.$$

The potential areas/services tactics depend on the idea of the area-service conformance relationship. Conformance between CS_j and A_i, notably $A_i \lhd CS_j$, means deploying CS_j over a cloud resource in A_i can totally satisfy the QoS requirement of CS_j. This relationship can be calculated via the current status of the area the service. However, for Type-I, Type-II, or Type-III services, the specific formula is completely different. For a Type-II or Type-III service, whether it is conformance with a certain area can be calculated directly. As stated previously, Type-II and Type-III are computation-intensive services, whose demands for the computing power of cloud resource is relatively fixed (e.g. 100 UCRs per second). Therefore, if the computing power of a resource in this area can meet the requirement, we can judge simply that area/service is consistent, otherwise inconsistent. However, as for a Type-I service, the requests should be handled in real time. When CS_j is running on A_i with a single thread, we take a random variable w to indicate the response time (T^{RST}) for each service request. According to M/M/1 queuing model [21], the probability distribution of each request can be responded within w will satisfy:

$$F(w) = P\{T^{RST} \le w\} = 1 - e^{-(\mu-\lambda)w},$$

where μ represents the speed each request of CS_j consumes, and λ represents the request arrival speed, and

$$\begin{cases} \mu = {V_i \cdot \delta_{ji}}/{\bar{D}_j} \\ \lambda = {\bar{v}_k^j}/{\bar{D}_j} \end{cases}$$

in which, $\bar{D}_j$ represents the amount of standard resource consumption for each execution of CS_j, V_i represents the speed of each resource in A_i, $\bar{v}_k^j$ is the average service request rate of CS_j during the renting period.

Suppose the probability of the response time exceeding T_j^{RST} for each request is not more than η. We will have

$$F\left(T_j^{RST}\right) = 1 - e^{-(\mu-\lambda)\cdot T_j^{RST}} \ge 1 - \eta.$$

If we substitute λ and μ, the consistency of A_i and CS_j should satisfy:

$$V_i \cdot \delta_{ji} \ge \bar{v}_k^j - \frac{\bar{D}_j \cdot \ln \eta}{T_j^{RST}}.$$

When we extend to the situation of running CS_j on A_i with multi-threads, the maximal service request rate which each thread in A_i can support would be

$$V_i \cdot \delta_{ji} + \frac{\bar{D}_j \cdot \ln \eta}{T_j^{RST}} \ge \bar{v}_k^j > 0.$$

We can finally conclude that a Type-I service CS_j is conformant with an area A_i *iff*

$$V_i \cdot \delta_{ji} > -\frac{\bar{D}_j \cdot \ln \eta}{T_j^{RST}},$$

and the number of empty threads is greater than

$$\left\lceil \frac{\bar{v}_k^j}{V_i \cdot \delta_{ji} + \frac{\bar{D}_j \cdot \ln \eta}{T_j^{RST}}} \right\rceil.$$

Denote the set for the cloud services that have made deals with some areas at time t as CS_{link}^t, then the number of the potential services for area A_i $\gamma_i^{ra}(t)$ is

$$\gamma_i^{ra}(t) = \left|\{CS_j | A_i \lhd CS_j\} - CS_{link}^t\right|.$$

Thus, suppose there are m services and n areas, when employing the polynomial function, A_i's asking function on the potential services tactic will be

$$a_i^{ra}(t) = a_i^{\min} + \left(a_i^{\max} - a_i^{\min}\right)\left(\frac{m - \gamma_i^{ra}(t)}{m}\right)^{\frac{1}{\beta}}.$$

And in like manner, CS_j's bidding function on the potential areas tactic will be

$$b_j^{ra}(t) = b_j^{\max} - \left(b_j^{\max} - b_j^{\min}\right)\left(\frac{|\{A_i | A_i \lhd CS_j\}|}{n}\right)^{\frac{1}{\beta}}.$$

If we further denote $\gamma_i^{rs}(t)$ as the number of empty threads (come from not been traded resources) in the area A_i, and $|A_i|$ as the number of total threads in A_i, then A_i's asking function on the remaining vacant resources will be

$$a_i^{rs}(t) = a_i^{\min} + \left(a_i^{\max} - a_i^{\min}\right)\left(\frac{|A_i| - \gamma_i^{rs}(t)}{|A_i|}\right)^{\frac{1}{\beta}}.$$

The overall asking and bidding prices for sellers and buyers will integrate the above-mentioned tactics. In this paper, we design two strategies to calculate the bid values for the services agents and the ask values for the cloud resources agents, i.e., static strategy and dynamic strategy. Specially, the static strategy assigns a static weight for each tactic. While the weights for different tactics by dynamic strategy vary with the remaining time. The twofold strategies are designed as follows.

1) *Static strategy*. Like [19] and [20], the services and resources agents' overall bidding and asking behavior is the amalgamation of the tactics. Conventionally, we can calculate the areas' asking price via

$$a_i(t) = \sigma^{rt} a_i^{rt}(t) + \sigma^{rs} a_i^{rs}(t) + \sigma^{ra} a_i^{ra}(t)$$

in which $\sigma^{rt}, \sigma^{rs}, \sigma^{ra}$ represent the weights for the remaining time, the remaining vacant resources and the potential services, respectively. To keep the dimensions between single and integrated asking tactics, the weights should satisfy:

$$0 \le \sigma^{rt}, \sigma^{rs}, \sigma^{ra} \le 1$$

and

$$\sigma^{rt} + \sigma^{rs} + \sigma^{ra} = 1.$$

In like manner, the services' bidding price will be

$$b_j(t) = \lambda^{rt} b_j^{rt}(t) + \lambda^{ra} b_j^{ra}(t)$$

in which $\lambda^{rt}, \lambda^{ra}$ are the weights of the remaining time and the potential area, respectively. Where

$$\lambda^{rt}, \lambda^{ra} \in [0, 1]$$

and

$$\lambda^{rt} + \lambda^{ra} = 1.$$

By static strategy, the weights of each tactic have nothing to do with the time and should be set according to the relative importance of different tactics by specific application.

2) *Dynamic strategy*. Making use of the dynamic asking strategy, A_i's asking price will be defined as

$$a_i(t) = \sigma^{rt}(t) \cdot a_i^{rt}(t) + \sigma^{rs}(t) \cdot a_i^{rs}(t) + \sigma^{ra}(t) \cdot a_i^{ra}(t)$$

in which

$$\begin{cases} \sigma^{rt}(t) = \frac{t - rp(t)}{\tau}, \\ \sigma^{rs}(t) + \sigma^{ra}(t) = 1 - \sigma^{rt}(t). \end{cases}$$

CS_j's bidding price will be

$$b_j(t) = \lambda^{rt}(t) \cdot b_j^{rt}(t) + \lambda^{ra}(t) \cdot b_j^{ra}(t),$$

where

$$\begin{cases} \lambda^{rt}(t) = \frac{t - rp(t)}{\tau}, \\ \lambda^{ra}(t) = \frac{\tau - (t - rp(t))}{\tau}. \end{cases}$$

By dynamic strategy, the agents will dynamically adjust the overall asking and bidding values. When the remaining time is sufficient, the sellers/bidders will pay more attention to other factors. The weights for remaining time tactic should be smaller, and the weights for other tactics should be bigger, since they have enough time to bargain. However, as time goes on, when there has little remaining time left, the sellers/bidders want to make a trade as soon as possible. Consequently, the weights for the remaining time tactic should be bigger.

3.3 Calculating the Final Dealing Price and Profit

For a cloud service CS_j and an area A_i, if their fitness value is δ_{ji}, and the asking and bidding prices are $a_i(t)$ and $b_j(t)$, the final dealing price is

$$p_{ij}(t) = \frac{b_j(t) \cdot \delta_{ji} + a_i(t)}{2}.$$

On this expression, we have some arguments about its properties:

1) Since $\delta_{ji} \geq 1$ commonly holds, there will usually be

$$\frac{b_j(t) \cdot \delta_{ji} + a_i(t)}{2} \geq \frac{b_j(t) + a_i(t)}{2}$$

i.e. for each cloud resource, resource provider for A_i will gain

$$b_j(t) \cdot \frac{\delta_{ji} - 1}{2}$$

more profit when employing FA instead of CDA.

2) When the expression

$$a_i(t) > (2 - \delta_{ji}) \cdot b_j(t)$$

holds, the final dealing price the resource provider receives may even be greater than the bidding price of the cloud service agent for CS_j, i.e.

$$p_{ij}(t) > b_j(t).$$

It seems that this conclusion might make the service agent for CS_j pay more than its bidding price for every A_i's cloud resource, and is also contradictive to the conclusion of the classical auction theory, in which

$$p_{ij}(t) \leq b_j(t)$$

always holds. However, as we learn from the discussion in Section 2.3 that the utility of A_i's resource is δ_{ji} times of the standard resource, the resource renter (service agent) thus pays according to its equivalent price instead of the dealing price. So, when a trade is executed between CS_j and A_i, the equivalent price for CS_j is

$$p'_{ij}(t) = \frac{p_{ij}(t)}{\delta_{ji}} = \frac{b_j(t) + a_i(t)/\delta_{ji}}{2}.$$

Since when the trade is executed, there is always be $b_j(t) \geq a_i(t)/\delta_{ji}$ holds, there will be

$$p'_{ij}(t) = \frac{b_j(t) + a_i(t)/\delta_{ji}}{2} \geq \frac{b_j(t) + b_j(t)}{2} = b_j(t).$$

Since $\delta_{ji} \geq 1$ commonly holds, it is easy to conclude that by using FA, both the resource providers and renters will gain more profit than the continuous double auction.

Furthermore, when

$$p'_{ij}(t) > a_i(t),$$

i.e.

$$b_j > a_i/\delta_{ji}$$

and

$$\delta_{ji} > 1.$$

A_i's dealing price p_{ij} will be greater than its asking price a_i, so that A_i will gain the extra profit

$$p_{ij}(t) - a_i(t) = \frac{b_j(t) \cdot \delta_{ji} - a_i(t)}{2}$$

for every traded cloud resource. Meanwhile, when

$$p'_{ij}(t) < b_j(t).$$

CS_j will also gain the extra profit

$$b_j(t) - p'_{ij}(t) = \frac{b_j(t) - a_i(t)/\delta_{ji}}{2}$$

for every traded cloud resource.

TABLE 1
Basic Parameters of Five Areas for the Experimental Environment

	A_1	A_2	A_3	A_4	A_5
Speed	12	40	24	20	36
Cost	25	25	25	25	30
Performance Traits	No accelerated	No accelerated	I/O accelerated	Multimedia accelerated	I/O accelerated
Number of Cloud Resource Threads	500	200	200	250	100

4 Experimental Study

This section evaluates the effectiveness of the CRAA/FA algorithm. In our experimental environment, there are five different areas, A_1, A_2, A_3, A_4 and A_5, whose speed, cost, performance traits and the number of resource threads are shown in Table 1. Meanwhile, we also deploy 60 cloud services with three different types, i.e. Type-I, Type-II and Type-III (we have described the definitions of the Type-I, Type-II and Type-III service in the *system model and cloud resource market* subsection of the *preliminaries* section), within which Type-I is of 75 percent, Type-II is of 20 percent, and Type-III is of 5 percent. These cloud services are further classed into three different service levels, with different budgets of 30, 50 and 80 respectively. Most of the cloud services' budgets are 30 and 50, while four of services have budgets of 80.

There will usually be $\beta = 1$ for the asking parameters of all these areas (from A_1 to A_5) unless the value of β is explicated. And when using the static asking (or bidding) strategies, the weights for $\sigma^{rt}(t)$, $\sigma^{rs}(t)$ and $\sigma^{ra}(t)$ will be 0.75, 0 and 0.25. While using the dynamic strategies, the following expressions will be used to calculate these weights

$$\begin{cases} \sigma^{rt}(t) = \frac{t - rp(t)}{\tau}, \\ \sigma^{rs}(t) = 0, \\ \sigma^{ra}(t) = \frac{\tau - (t - rp(t))}{\tau}. \end{cases}$$

As for the cloud services, we suppose the static bidding parameters on the remaining time tactic and the potential areas tactic are 2/3 and 1/3, respectively.

First, we experiment on the asking prices of five areas, either employing static or dynamic asking/bidding strategies. Both Figs. 3 and 4 present the variation of asking prices from the resource agents of A_1, A_2, A_3, A_4 and A_5. The difference is that Fig. 3 employs the dynamic strategies, while Fig. 4 employs the static one. It is shown that as the remaining time decreases, all of the agents reduce their asking prices for their resources, and reach or approach the cost of their resources. When some trades are executed, a sudden fluctuation occurs since the number of available services reduced. When employing the dynamic strategies, the asking prices decrease slowly at beginning, while becoming faster and faster as the time goes by. However, when employing the static strategies, the asking prices decrease at a constant speed.

To get a more precise evaluation of the efficiency of the CRAA/FA algorithm, we formally define some measures, including the dealing rate, the average fitness, the average fit rate, the average premium rate, and the total interests.

The *dealing rate* is the percentage of the cloud services that finally make deal among all the cloud services. We suppose CS_j finally makes deal with a certain area for the kth renting period, the fitness between CS_j and that area is δ_j, and the amount of standard resource CS_j consumes for every request is D_j. If we further suppose $\delta_j = 0$ when CS_j fails to make deal with any resources, then the *average fitness* is:

$$\overline{\delta}^k = \frac{\sum_{j=1}^{m} \delta_j D_j}{\sum_{j=1}^{m} D_j}$$

Furthermore, suppose $\hat{\delta}_j$ be the maximum element in CS_j's fitness vector, the upper bound of the average fitness of the whole market is:

$$\frac{\sum_{j=1}^{m} \hat{\delta}_j D_j}{\sum_{j=1}^{m} D_j},$$

and the *average fit rate* for a certain bargaining result will be:

$$\frac{\sum_{j=1}^{m} \delta_j D_j}{\sum_{j=1}^{m} D_j} \Big/ \frac{\sum_{j=1}^{m} \hat{\delta}_j D_j}{\sum_{j=1}^{m} D_j} = \frac{\sum_{j=1}^{m} \delta_j D_j}{\sum_{j=1}^{m} \hat{\delta}_j D_j}.$$

From the definitions above, we can see clearly that the average fitness well reflects the change of resource

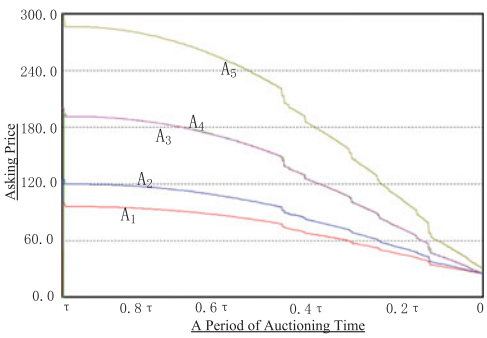


Fig. 3. Variation of asking prices from the resource agents via dynamic strategies.

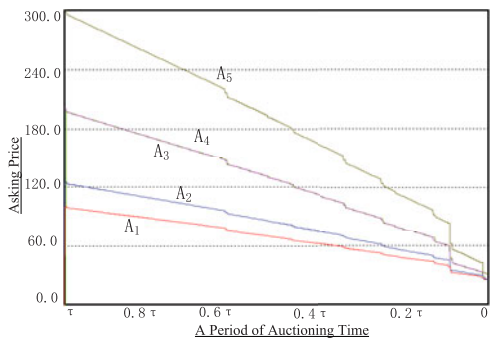


Fig. 4. Variation of asking prices from the resource agents via static strategies.

TABLE 2
Values of the Measures for the Dynamic Asking/Bidding Strategies

r	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
Dealing Rate %	100	100	100	100	100	100	100	100	100	100
Average Fitness	2.914	2.914	2.914	2.914	2.914	2.914	2.914	2.914	2.914	2.914
Average Fit Rate %	99.13	99.13	99.13	99.13	99.13	99.13	99.13	99.13	99.13	99.13
Average Premium Rate %	179.77	187.38	192.31	195.28	198.71	200.53	203.54	204.82	206.73	207.94
Total Interests	15,356	16,581	17,308	17,865	18,730	18,949	19,412	19,758	19,958	20,346

allocation efficiency when increasing or decreasing cloud resources to certain areas. While, when the amount of cloud resources in each area is identified, the average fit rate can reflect the impacts of current bidding parameters on resource allocation efficiency, and the average fit rate is a value between 0 and 1 and usually close to 1. Based on the discussion, we learn that both the average fitness and the average fit rate actually reflect in what extent a cloud service is able to get its suitable resource. Moreover, since we define $\delta_j = 0$ when CS_j fails to make deal with any resources, they also indirectly reflect the dealing status of the market. So, we can conclude that the average fitness and the average fit rate are two of the most important measures showing the overall resource allocating efficiency for the whole cloud resource market.

Moreover, if the dealing price between A_i and CS_j is p_j, the amount is x_j UCRs, and the cost of every unit of standard cloud resource in area A_i is $c_A(j)$, then for the kth renting period the *average premium rate* is

$$\frac{\sum_{j=1}^{m} p_j \cdot x_j}{\sum_{j=1}^{m} c_A(j) \cdot x_j}$$

and the *total interests* is

$$\sum_{j=1}^{m} p_j \cdot x_j - \sum_{j=1}^{m} c_A(j) \cdot x_j.$$

Based on the above measures, we set the cost of every cloud resource in area A_i be c_i, $a_i^{\min} = c_i$, $a_i^{\max} = k_i r$, and (k_1, k_2, ..., k_5) = (50, 62.5, 100, 100, 150). We vary the value of r from 1.2 to 3.0, with a step value of 0.2. Table 2 presents the values of those measures when applying the dynamic strategies, and Table 3 presents the values when applying the static strategies.

As can be seen from Tables 2 and 3: 1) the matching efficiency (concerning average fitness and the average fit rate, which can guarantee the performance of cloud services) of the resource allocation for the dynamic strategy proposed in this paper is similar with the static strategy. 2) the increment of r has little influence on the matching efficiency for both of the dynamic and static strategy. 3) the market efficiency (concerning average premium rate and total interests) by dynamic strategy is slightly worse than static strategy.

When we further enlarge the value of r, we have some new interesting findings. As can be seen form Figs. 5 and 6, the market efficiency (concerning dealing rate and the market's total interests) by dynamic strategy gradually show obvious advantages as the r value increases from 1.5 to 200, when compared with the static strategy. This observation indicates that the range of r in [3, 200] can more obviously reflect the advantage of dynamic strategy. From Figs. 5 and 6, we can also get other two observations: 1) when the value of r is out of the reasonable range, the dealing rate drops as r increases. 2) the total interests of the whole market will not always increase with the growth of the maximum asking price (reflected by r value), it will fall when r is unrealistically high. These observations are reasonable in traditional continuous double auction issues.

In sum, when r is small, there appears little changes between the experimental results. On the contrary, when r is larger, the results change in the larger. In practice, the r should be set with a reasonable value according to different management objective of specific cloud resource allocation market.

We will now turn to the issue of the increment of the resource quantity versus the allocation efficiency. In addition to such measures as the total interests, a new measure, *the market share*, is introduced in this topic. Suppose $Deal_k$ is the cloud service set, in which each element is a cloud service that has successfully made deal with a certain area in the kth renting period, $Deal_k(i)$ is the cloud service set traded with area A_i in the kth renting period, and the amount of standard resource CS_j consumes for every request is D_j. Thus,

$$\frac{\sum_{CS_j \in Deal_k(i)} D_j}{\sum_{CS_j \in Deal_k} D_j}$$

will be the market share of A_i in the kth renting period.

Let the quantity of cloud resources in A_1 and A_2 increase from 0 to 10,000 respectively, when employing the dynamic strategies, the variation of market shares for each area are

TABLE 3
Values of the Measures for the Static Asking/Bidding Strategies

r	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
Dealing Rate %	100	100	100	100	100	100	98.33	86.67	86.67	86.67
Average Fitness	2.914	2.914	2.914	2.914	2.914	2.914	2.914	2.912	2.912	2.912
Average Fit Rate %	99.13	99.13	99.13	99.13	99.13	99.13	99.13	99.07	99.07	99.08
Average Premium Rate %	181.03	188.70	194.88	197.97	202.2	206.27	211.67	219.68	222.94	225.24
Total Interests	14,807	16,211	18,074	19,569	20,374	20,563	20,994	21,872	22,190	22,763

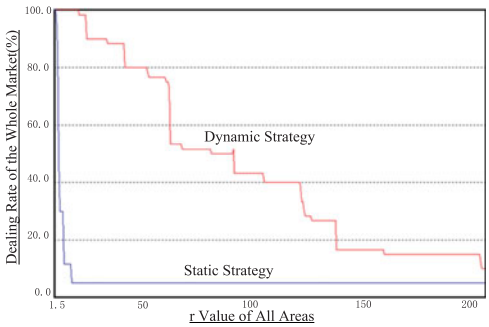


Fig. 5. Variations of the dealing rate as the r value increases from 1.5 to 200.

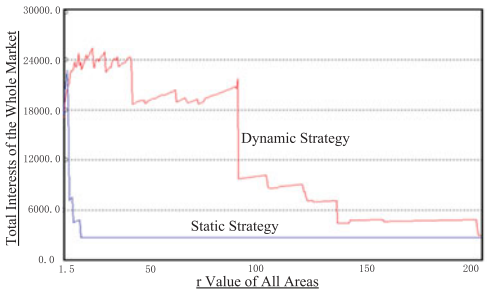


Fig. 6. Variations of the total interests as the r value increases from 1.5 to 200.

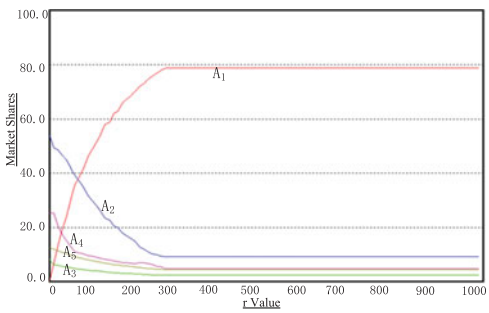


Fig. 7. Variations of the market shares of each area as the quantity of cloud resource in A_1 increases from 0 to 10,000.

shown in Figs. 7 and 8, respectively. And Fig. 9 shows the variation of total interests for each area as the quantity of cloud resources in A_1 increases from 0 to 10,000.

We learned from Table 1 that A_1 is the lowest in the speed amongst these areas, and it is also not I/O accelerated or multimedia accelerated. However, as is shown in Fig. 7, its market increases as the quantity of cloud resources in A_1 increases, until it reaches a certain level. In the mean while, the market shares of the other areas decreases as A_1 increases, until they all reach certain levels.

After a careful analysis of the experiment related to Figs. 7, 8 and 9, we can arrive at the following conclusions of our cloud resource markets:

1) Since most of services are Type-I and their resource consumption and requirements are limited, they tend to rent cheaper resources with only suitable performance traits. In Fig. 9, since cloud resources in A_1 are often cheaper than the other areas (shown in Fig. 3), and its performance traits satisfy most of these Type-I services, the cloud services that used to rent other resources will switch to rent A_1's resources as the quantity in A_1 increases. Comparing the allocation results, we can see that the number of cloud services that trade with A_2, A_3, A_4 decrease from 29, 8, 17 to 15, 8, 13 as the number of cloud resource increase from 200 to 2,000, and the decreased services all trade with A_1 at last.
2) When the market share of a certain area reaches its peak, this area actually has already made deal with all of cloud services that are suitable with its performance traits. In another word, this cloud area has taken all of the potential market shares it could, with the help of its low prices, so that keeping increasing its quantity of cloud resources wont help increase its market share. For example, as we can see from Fig. 7, the market share of A_1 stops increasing as soon as its cloud resource reaches 2,700, which means A_1 has, from then on, made deal with all services that are satisfied with its performance traits, and as for the rest services, the performance of A_1 cannot satisfy their QoS requirements. The same thing also happens to A_2 in Fig. 8. Moreover, as is shown in Fig. 9, after the cloud resource exceed 2,700, the total interest of A_1 actually decreases as the quantity of resources continues to grow. This is because A_1's asking prices begin to decrease after 2,700, while its market share remains still.

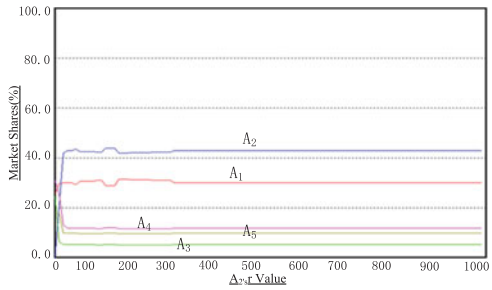


Fig. 8. Variations of the market shares of each area as the quantity of cloud resource in A_2 increases from 0 to 10,000.

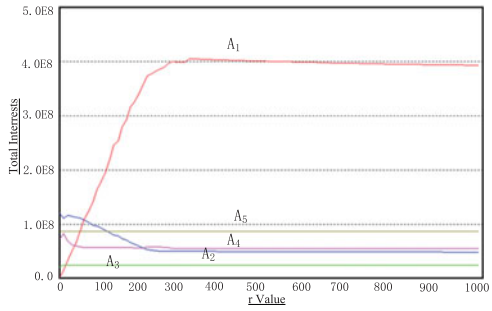


Fig. 9. Variations of the total interests of each area as the quantity of cloud resource in A_1 increases from 0 to 10,000.

We now experiment on the impact of the cloud resource supply to the distributions of both areas and services. Suppose s_i is the number of cloud resource, whose value is specified as Table 1. At the beginning of the experiment, we set the resource quantity of all the areas to 0. And during the experiment, these quantities increase with $s_i * 100$ for every step, until they arrive at s_i. The dealing rate variations of the services are shown in Fig. 10, and the areas are shown in Fig. 11.

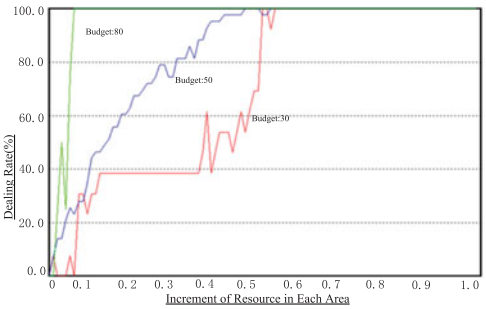


Fig. 10. Dealing rates variations for cloud services as the resources increases.

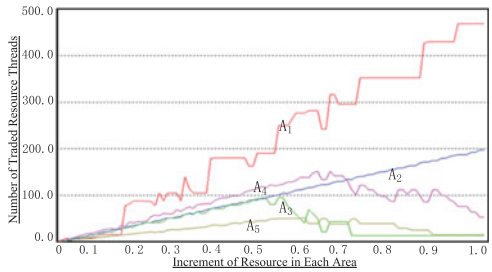


Fig. 11. Variations on the number of traded resources for cloud areas as the resources increases.

In all, as is shown in Fig. 10, when the resources of all areas A_i increase from 0 to s_i, the dealing rate for cloud services with different QoS are all increasing. At the beginning while the overall cloud resources are scarce, the cloud service with a higher service level (having more budget) is more likely to make a deal. Later, when there are enough resources, all of these services are able to acquire their needed resources.

As for the areas, we see from Fig. 11 that the newly added resources will immediately be allocated at the beginning when the overall cloud resource is insufficient, so that every area at that time is almost 100 percent traded. Later on, as the overall quantity of resources become sufficient, the dealing rates of A_3, A_4 and A_5 start dropping. However, A_1 and A_2 are still increasing due to their low asking prices. Based on these observations, it is easy to conclude that the increments of A_1 and A_2 are achieved actually by taking away the market shares of A_3, A_4 and A_5, and this is the reason why the dealing rates of A_3, A_4 and A_5 start dropping after the market saturation.

To show that the introduction of fitness does improve the overall efficiency in the cloud resource allocation market, we will compare the values of some critical measures in the following experiments, including the dealing rate, the resource using rate, the vacancy rate of the Type-I services, and the average premium rate. Suppose CS_k^{deal} is the set of cloud services that have made deals with certain areas, $CS_j \in CS_k^{deal}$, A_i is the area that makes deal with CS_j, notably $CS_j \xrightarrow{k} A_i$, and they have made the transactions for N_{ji} units of standard cloud resource in A_i. Thus, the market's *resource using rate* for the kth renting period is

$$\frac{\sum_{CS_j \in CS_k^{deal}, CS_j \xrightarrow{k} A_i} \delta_{ji} V_i N_{ji}}{\sum_{CS_j \in CS_k^{deal}, CS_j \xrightarrow{k} A_i} V_i N_{ji}}.$$

We further suppose $CS_k^{deal}(I)$ is a subset of CS_k^{deal}, in which each element is a Type-I service, and $CS_j \in CS_k^{deal}(I)$. If x_j^{demand} is the minimum amount of units of standard cloud resource in A_i so as to meet with the QoS requirements of CS_j, the *vacancy rate of the Type-I services* in the kth renting period is

$$\frac{\sum_{CS_j \in CS_k^{deal}(I), CS_j \xrightarrow{k} A_i} (V_i N_{ji} - x_j^{demand})}{\sum_{CS_j \in CS_k^{deal}(I), CS_j \xrightarrow{k} A_i} V_i N_{ji}}$$

in which

$$x_j^{demand} = \frac{1}{\delta_{ji}} \left(\frac{\bar{v}_k^j}{h_j} - \frac{\bar{D}_j \cdot \ln \eta}{T_j^{RST}} \right).$$

Table 4 shows the market efficiency comparisons for the CRAA/FA algorithm and the algorithms without fitness like those introduced in [6], [11], [12], presenting the experiment data of the above 4 measures. We can conclude from these data that: 1) It is possible for CRAA/FA to arrive at the 100 percent dealing rate (in the buyers' market), i.e. all the services are able to get the required resources when there are enough resource overall. However, when employing those without fitness, the dealing rate decreases to 91.67 percent. 2) Fitness greatly increases the overall resource using rate from 1.01 to 1.17. The using rate of 1.01 actually means no accelerator has ever taken real effects. By introducing fitness, it increases by 15.84 percent. 3) Fitness decreases the vacancy rate of Type-I services and increases the average premium rate, which means it really increases the allocation efficiency, and makes both the resource providers and renters benefit from the auction.

5 Related Work

As far as the research on the resource allocation problem under the cloud environment is concerned, it in one hand follows the open issues discussed in the distributed and grid computing, like the prediction and reservation of resources [22], resource allocation based on SLA [23], and the workflow-based allocations [24]. On the other hand, researchers also begin to consider new features under the cloud environment, including the allocations via auctioning and the market mechanism [11], [12], [25], [26], virtual machine (VM) allocation [22], [23], [24], VM on-demand

TABLE 4
Market Efficiency Comparisons for Resource Allocation Algorithms with and without Fitness

	Dealing Rate	Resource Using Rate	Vacancy Rate of Type-I	Average Premium Rate
With Fitness	100%	1.17	3.00%	196.33%
Without Fitness	91.67%	1.01	3.19%	162.13%
Improving Rate %	9.09%	15.84	5.96%	21.08%

allocation [22], [23], [27], [28], live migration [29] and security-aware allocation [30].

From the works above, we can see some features regarding to the research of the resource allocation in cloud computing. First of all, a number of researches use virtual machines as their scheduling units, like [22], [23], [24], [29]. Moreover, they also prefer Amazon EC2 [31] or Eucalyptus [32], an open source implementation of Amazon Web services, as their experimental environments. This might due to the fact that it is easier to rent real cloud resources for experiments, and most of today's cloud services in use are IaaS. Secondly, many of these researches try to use the specific traits of elastic computing cloud like on-demand provision and live migration. One example of these work is [23], whose discussion of the service level agreement under the cloud environment is totally based upon these traits. In some sense, these specific traits actually reflect one kind of special need in cloud resource allocation. At last, these works notice that different parties in the cloud environment usually have different commercial interests, and propose several auction-based resource allocation methods like [11], [12], [25], [26]. However, although the ideas of auction-based allocation have been investigated under traditional distributed computing and grid computing [6], [33], they merely regard the auction method as a means to achieve load balancing among the computing nodes, other than a means to achieve a equilibrium point among different business parties.

Auction-based resource allocation methods in cloud computing are actually game-theoretic approaches, which receive much attention in the recent years. These methods all assume a resource market between resource providers (sellers) and users (buyers), and arrive at Nash equilibrium via the invisible hand in the market [6], [11], [12], [25], [26], [33]. Amongst the auction-based approaches, continuous double auction is often used [6], [8], [9], [10], [11], [12] as it takes many advantages over other auctioning methods like the English auction [4], [5].

In earlier years, Ferguson [33] proposed a method to map the elements in the computing resource allocation problem into the market model, and designed a resource allocation and control algorithm based on the auction approach. Later, Jennings et al. addressed and compared different bidding strategies for a bidding agent, and presented the ideas behind these strategies in their works [19] and [20]. The idea behind as well as the strategies are later referenced by many works including [6], [11], [12] and our work. Izakian et al. [6] carefully analyze the market model in grid computing, further presenting and experimenting on their CDA-based resource allocation method. Lin et al. [11] introduced a similar method in the cloud computing, however, they further discussed the value variation and revenue inferiority problem with the peak/off-peak concept. Sun et al. [12] also employs the CDA-based method, however, they proposed a resource allocation model that tries to deal with the performance-QoS and economic-QoS problem in cloud computing.

In cloud computing industry, as for Hadoop YARN [13] and Apache MESOS [14], they are both widely-used resource allocating frameworks. Hadoop YARN initiates from the JobTracker component of Hadoop 1.x, which comprises both a resource allocating and a task scheduling module, and becomes an independent software component since the release of Hadoop 2.0. Apache MESOS is also a resource management framework for server clusters, which was started at UC Berkeley in 2009, and is now graduated as a top level project of Apache. It is in production use at many companies including Twitter and AirBnB.

6 Conclusions and Future Work

In this paper, we introduce a new auction method, deal with the special challenges in the cloud computing to distribute the resources to the proper cloud services so that services are able to get their suitable resources, and services with higher service level are easier to get high quality resources. This method features the fitness concept, and the redesigned bargaining process and function to calculate the final dealing price. In this way, the overall market efficiency is fully improved. Experiments validate the CRAA/FA algorithm, and show that it is more efficient than the allocating methods without the introduction of fitness.

The future work is going to be carried out regarding to the scale problem of this allocation algorithm. Since *cloud* is usually serving thousands of users, a cloud composed of thousands of, or even tens of thousands of, servers are quite common. However, due to the limitation of our research funding, it is completely impossible to experiment under an environment with more than a thousand servers or virtual machines. To investigate the scalability of this allocation algorithm and compare with decentralized auction-based cloud resource allocation approaches, e.g., catallaxy-based approach [18], we are planning to rewrite some software modules so that it will be adapted to the environment with some VMs deployed in our own servers and others in a public IaaS cloud like Amazon EC2. By this means, we are able to setup an experimental environment with around 1,000 virtual machines.

Another aspect that is worth investigating is the oligopoly or even monopoly problem in our cloud resource market. In fact, we simply consume that cloud resources are located in a competitive resource market in our work. An oligopoly cloud resource market is the one in which all cloud resources, or a special type of cloud resource, like the one with high I/O throughput, are controlled by only a few providers. Can the CRAA/FA algorithm be used in such a cloud resource market? Are there more efficient resource allocation methods? We will continue to investigate that in our future work.

Acknowledgments

This work is partially supported by NSFC Key Project (No. 61232007) and Doctoral Fund of Ministry of Education of China (No. 20120092110028).

References

[1] G. Lin, D. Fu, J. Zhu, and G. Dasmalchi, "Cloud computing: IT as a service," *IT Prof.*, vol. 11, no. 2, pp. 10–13, 2009.

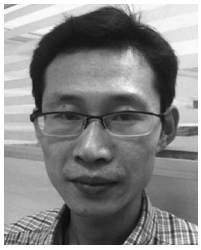
[2] S. Srinivasan and V. Getov, "Navigating the cloud computing landscape-technologies, services, and adopters," *Comput.*, vol. 44, no. 3, pp. 22–23, Mar. 2011.

[3] A. Velte, T. Velte, and R. Elsenpeter, *Cloud Computing: A Practical Approach*. New York, NY, USA: McGraw-Hill, 2010.

[4] H. Ma, *Bidding Strategies in Agent-Based Continuous Double Auctions*. New York, NY, USA: Springer Science & Business Media, 2008.
[5] M. Posada, C. Hernndez-Iglesias, and A. López-Paredes, "Learning in continuous double auction market," in *Artificial Economics*. Berlin, Germany: Springer, 2006.
[6] H. Izakian, A. Abraham, and B. Ladani, "An auction method for resource allocation in computational grids," *Future Gener. Comput. Syst.*, vol. 26, no. 2, pp. 228–235, 2010.
[7] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *Concurrency Computation: Practice Experience*, vol. 14, nos. 13-15, pp. 1507–1542, 2002.
[8] Y. Lan, W. Tong, Z. Liu, and Y. Hou, "Multi-unit continuous double auction based resource allocation method," in *Proc. 3rd Int. Conf. Intell. Control Inf. Process.*, 2012, pp. 773–777.
[9] P. Bonacquisto, G. D. Modica, G. Petralia, and O. Tomarchio, "A strategy to optimize resource allocation in auction-based cloud markets," in *Proc. IEEE Int. Conf. Services Comput.*, 2014, pp. 339–346.
[10] R. Prodan, M. Wieczorek, and H. Fard, "Double auction-based scheduling of scientific applications in distributed grid and cloud environments," *J. Grid Comput.*, vol. 9, no. 4, pp. 531–548, 2011.
[11] W. Lin, G. Lin, and H. Wei, "Dynamic auction mechanism for cloud resource allocation," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, 2010, pp. 591–592.
[12] D. Sun, G. Chang, C. Wang, Y. Xiong, and X. Wang, "Efficient Nash equilibrium based cloud resource allocation by using a continuous double auction," in *Proc. Int. Conf. Comput. Des. Appl.*, 2010, pp. 94–99.
[13] Hadoop yarn [Online]. Available: http://hadoop.apache.org, 2015.
[14] Apache mesos [Online]. Available: http://mesos.apache.org, 2014.
[15] Z. Kang and H. Wang, "A novel approach to allocate cloud resource with different performance traits," in *Proc. IEEE Int. Conf. Services Comput.*, 2013, pp. 128–135.
[16] D. Grosu and A. Das, "Auction-based resource allocation protocols in grids," in *Proc. 16th Int. Conf. Parallel Distrib. Comput. Syst.*, 2004, pp. 20–27.
[17] T. Eymann, M. Reinicke, W. Streitberger, O. Rana, L. Joita, D. Neumann, B. Schnizler, D. Veit, O. Ardaiz, P. Chacin, I. Chao, F. Freitag, L. Navarro, M. Catalano, M. Gallegati, G. Giulioni, R. C. Schiaffino, and F. Zini, "Catallaxy-based grid markets," *Multiagent Grid Syst.*, vol. 1, no. 4, pp. 297–307, 2005.
[18] O. Ardaiz, P. Artigas, T. Eymann, F. Freitag, L. Navarro, and M. Reinicke, "The catallaxy approach for decentralized economic-based allocation in grid resource and service markets," *Appl. Intell.*, vol. 25, no. 2, pp. 131–145, 2006.
[19] P. Faratin, C. Sierra, and N. Jennings, "Negotiation decision functions for autonomous agents," *Robot. Auton. Syst.*, vol. 24, no. 3, pp. 159–182, 1998.
[20] P. Anthony and N. Jennings, "Developing a bidding agent for multiple heterogeneous auctions," *ACM Trans. Internet Technol.*, vol. 2, no. 3, pp. 185–217, 2003.
[21] L. Kleinrock, *Queuing Systems, Volume Two: Computer Application*. New York, NY, USA: Wiley, 1976.
[22] S. Chaisiri, L. B.S., and D. Niyato, "Optimization of resource provisioning cost in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 164–177, Apr.–Jun. 2012.
[23] K. Prasad, T. Faruquie, L. Subramaniam, and M. Mohania, "Resource allocation and SLA determination for large data processing services over cloud," in *Proc. IEEE Int. Conf. Services Comput.*, 2010, pp. 522–529.
[24] T. Huu and J. Montagnat, "Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, 2009, pp. 612–617.
[25] F. Teng and F. Magoules, "Resource pricing and equilibrium allocation policy in cloud computing," in *Proc. 10th IEEE Int. Conf. Comput. Inf. Technol.*, 2010, pp. 195–202.
[26] B. An, V. Lesser, D. Irwin, and M. Zink, "Automated negotiation with decommitment for dynamic resource allocation in cloud computing," in *Proc. 9th Int. Conf. Auton. Agents Multiagent Syst.*, 2010, pp. 981–988.
[27] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.
[28] P. Pillai and S. Rao, "Resource allocation in cloud computing using the uncertainty principle of game theory," *IEEE Syst. J.*, vol. PP, no. 99, pp. 1–12, 2014, Doi: 10.1109/JSYST.2014.2314861.
[29] J. Li, M. Qiu, and J. Niu, "Adaptive resource allocation for preemptable jobs in cloud systems," in *Proc. 10th Int. Conf. Intell. Syst. Des. Appl.*, 2010, pp. 31–36.
[30] S. A.H., E. A.S., and R. H. V., "Security-aware resource allocation in clouds," in *Proc. IEEE Int. Conf. Services Comput.*, 2013, pp. 400–407.
[31] Amazon elastic compute cloud (amazon ec2) [Online]. Available: http://aws.amazon.com/ec2, 2014.
[32] Eucalyptus [Online]. Available: http://open.eucalyptus.com, 2015.
[33] P. Pillai and S. Rao, "Resource allocation in cloud computing using the uncertainty principle of game theory," *IEEE Syst. J.*, pp. 1–12, 2014, Doi: 10.1109/JSYST.2014.2314861.

Hongbing Wang received the PhD degree in computer science from Nanjing University, China. He is a professor in the School of Computer Science and Engineering, Southeast University, China. His research interests include service computing, cloud computing, and software engineering. He published more than 50 refereed papers in international Journals and conferences. He is a member of the IEEE.

Zuling Kang received the PhD degree in 2013 from the School of Computer Science and Engineering, Southeast University, China. He is currently an architect on distributed and big data systems in China Mobile Group Zhejiang Co., Ltd. His research interests include cloud computing, big data systems, service computing, and distributed database systems. His publications mainly appeared in international journals and popular conferences.



Lei Wang is currently (2010-present) working toward the PhD degree in the School of Computer Science and Engineering, Southeast University, PR China. He is also a lecturer in the Nanjing Forestry University, PR China. His research interests include service computing, software reliability engineering, and data mining.