IEICE TRANS. INF. & SYST., VOL.E95–D, NO.1 JANUARY 2012

PAPER

Combinatorial Auction-Based Marketplace Mechanism for Cloud Service Reservation

Ikki FUJIWARA[†a)], **Kento AIDA**[†,††], ***Members***, ***and*** **Isao ONO**[†††], ***Nonmember***

SUMMARY This paper proposes a combinatorial auction-based marketplace mechanism for cloud computing services, which allows users to reserve arbitrary combination of services at requested timeslots, prices and quality of service. The proposed mechanism helps enterprise users build workflow applications in a cloud computing environment, specifically on the platform-as-a-service, where the users need to compose multiple types of services at different timeslots. The proposed marketplace mechanism consists of a forward market for an advance reservation and a spot market for immediate allocation of services. Each market employs mixed integer programming to enforce a Pareto optimum allocation with maximized social economic welfare, as well as double-sided auction design to encourage both users and providers to compete for buying and selling the services. The evaluation results show that (1) the proposed forward/combinatorial mechanism outperforms other non-combinatorial and/or non-reservation (spot) mechanisms in both user-centric rationality and global efficiency, and (2) running both a forward market and a spot market improves utilization without disturbing advance reservations depending on the provider's policy.

key words: *cloud computing, resource allocation, combinatorial auction, integer programming, optimization*

1. Introduction

Building enterprise systems on a cloud computing platform is becoming increasingly popular these days. In contrast to a conventional on-premise computing system, to which the user has to invest in dedicated hardware and software, the cloud computing system delivers virtualized hardware and software resources to users on demand via the internet, generally in a pay-per-use manner. It significantly reduces the cost for deploying and maintaining enterprise systems.

The cloud is described as a three-tier structure, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) from low-layer to high-layer. Recently, PaaS has been evolving rapidly as a software development/deployment environment for enterprise systems. For example, Microsoft Windows Azure [1] provides .NET development environment and SQL service, whereas Google App Engine [2] provides Python and Java development environment with key-value store service. A developer chooses an appropriate service among available ones to build his customized system.

An enterprise system is generally consists of multiple subsystems to model a complex real business. The subsystems are often provided by service on the internet, or PaaS, and the developer of the enterprise system needs to choose appropriate PaaS providers to develop an efficient system. As the number of PaaS provider will increase, a challenging issue is how to choose an appropriate combination of services, or PaaSes, to build a complex enterprise system. Furthermore, the enterprise system has strict requirements for the quality of service (QoS) as well as a budget limitation. Unfortunately no practical workaround to this complex problem has been provided so far in the cloud computing environment.

This kind of problem has long been discussed as a resource allocation problem on a distributed computing system [3]. The problem is typically described as a sort of optimization problem and tends to have computational complexity to obtain optimum solutions. A market-based approach is a promising methodology to deal with the complexity while satisfying budget limitation [4], [5]. Specifically a combinatorial auction-based approach has recently evolved because of its ability to optimize allocation of a bundle of multiple goods [6], [7]. However, the previous work, not only market-based approaches but also conventional ones, does not satisfy requirements of resource allocation in the enterprise system composed of multiple PaaSes.

In this paper, we propose a combinatorial auction-based marketplace mechanism for cloud computing services, which allows users to reserve arbitrary combination of services at requested timeslots, prices and quality of service. The proposed mechanism helps enterprise users build workflow applications on PaaS, where the users need to compose multiple types of services at different timeslots. The proposed marketplace mechanism consists of a forward market for an advance reservation and a spot market for immediate allocation of services. Each market employs mixed integer programming to enforce a Pareto optimum allocation with maximized social economic welfare, as well as double-sided auction design to encourage both users and providers to compete for buying and selling the services.

The rest of this paper is organized as follows. Section 2 presents the background and the related work, and Sect. 3 shows cloud computing model discussed in this paper. Section 4 shows the proposed market mechanism. Section 5 presents our simulator and Sect. 6 shows the performance evaluation of the proposed mechanism. Finally, Sect. 7 sum-

Manuscript received May 16, 2011.
Manuscript revised September 6, 2011.

[†]The authors are with the Graduate University for Advanced Studies (SOKENDAI), Tokyo, 101–8430 Japan.
[††]The author is with National Institute of Informatics, Tokyo, 101–8430 Japan.
[†††]The author is with Tokyo Institute of Technology, Yokohama-shi, 226–8502 Japan.
a) E-mail: ikki@nii.ac.jp

DOI: 10.1587/transinf.E95.D.192

marizes our contributions and outlines the future work.

2. Background and Related Work

Our work is mainly related to two research areas: (1) auction theory and (2) distributed computing. Below we review the previous work in these areas.

2.1 Combinatorial Auction

The backbone of a market mechanism is the auction theory. The type of auction varies with the number of goods and the side of price decision [8]. Among them the double-sided combinatorial auction is the most generic form of auctions, where the participant can bid on a combination of multiple goods and both of the sellers and buyers express their valuation [9]. It is well known that the combinatorial auction is NP-complete. Theoretical researches of combinatorial auction therefore tend to focus on approximate algorithms rather than exact algorithms [10]–[13].

We modeled our service allocation problem as a combinatorial auction. The combinatorial auction is suitable to simulate our target model, where each user tries to reserve multiple services to build the user's application, or the enterprise system. We employ MIP-based exact algorithm to solve the allocation problem. Currently, fast software tools to solve MIP in practical time are available [14], [15]. Our preliminary experiments indicate that computation time to solve the MIP for our target model is acceptable [16].

2.2 Distributed Resource Allocation

Market-based resource allocation in a distributed computing environment is discussed over a decade. Buyya et al. provides some comprehensive surveys in this area [17]–[20] as well as grid/cloud computing toolkits and simulators [21], [22]. The researches in this area mainly focus on a fair use of academic computing resources. Spawn [23] by Waldspurger et al. is the first market-based distributed resource allocation method. It employs distributed auctions with double-sided competition. The user is required to modify his program to take part in Spawn system. Nimrod/G [24] by Buyya et al. is a negotiation-based grid scheduler built on the top of Globus [25]. The allocation and the price are determined on negotiations between a provider and a user, rather than auctions. Bellagio [26] by AuYoung et al. proposes a centralized market of shared resources. It employs single-sided combinatorial auctions where the providers are out of competition. Tan et al. [27] proposes a stable continuous double auctions (SCDA) which emulates combinatorial auctions by doing single-good auctions repeatedly. Schnizler et al. [28] introduced the notion of using double-sided combinational auctions to allocate grid resources to the user's application presented by workflow. It employs a mixed integer programming to obtain exact solution rather than a heuristics. We thought this might be suitable to the cloud marketplace; however the users cannot combine arbitrary resources in different timeslots to compose a workflow.

Conventional resource management systems for distributed computing, such as PBS [29], SGE [30] and Condor [31], are not based on an auction mechanism, therefore are not comparable with the proposed mechanism. Furthermore, these conventional systems are aimed to queue jobs for available resources in cluster/grid computing environment, and are not designed to reserve combination of services in cloud computing environment. To the best of our knowledge, no conventional system supports both the auction-based allocation and the enterprise cloud application.

Cloud computing industries have developed some kind of marketplace to promote their services and extend their ecosystem. Amazon EC2 Spot Instances [32] enable the provider to change the price of his IaaS and the users to bid for it. Heroku Add-ons [33] enables the provider to sell his PaaSes at a posted price and the users to buy them as a component of their applications. However, we have so far noticed no double-sided auction system for cloud computing services in production.

As mentioned above, marketizing resource allocation is the wave of the future computing systems, whereas the adoption of auction mechanism by cloud computing industry is still in its infancy. Further research is needed to establish an efficient market mechanism meeting the requirements of the enterprise cloud applications.

2.3 Other Disciplines

Electricity markets are in practical operation for several years. For instance, Japan Electric Power Exchange (JPEX) started operations in 2005. According to Ref. [34], it provides three markets: (1) a spot market for trading the electricity on the next day, (2) a forward market for trading the electricity to be delivered weeks or months ahead and (3) a forward bulletin board market for free transactions. Since electricity and computing services have similar natures (i.e. they cannot be stored), we regard the electricity market as a preceding model to the cloud services market. However, the electricity market model cannot be directly applied to cloud computing because the electricity is almost uniform, whereas computing services vary in type and quality.

The stock market deals with a variety of stocks, which can be stored and resold, unlike a computing service. The studies on dealing strategies and mechanism design have used multi-agent simulations. U-Mart [35] is a test bed for multi-agent simulations of the stock market, and it is especially focused on futures trading. It allows machine agents and human agents to trade future stocks at the same time. Our simulator presented in this paper is developed with an interface with the U-Mart system, so that we can evaluate performance of various machine/human agents.

3. Cloud Computing Model

This section presents the cloud computing model discussed in this paper. In this model, users build their applications, or enterprise systems, by reserving multiple services offered by providers. The marketplace brokers services between users and providers.

3.1 Cloud Service

The "cloud" spreads over a wide level of abstraction from hardware to software. Now it is understood in a three-tier structure from low-level to high-level: Infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a service (SaaS). Many providers compete in each tier for selling their own services, making it increasingly difficult for users to select an appropriate service among them.

In this paper, we assume that an enterprise system is implemented using PaaS. PaaS is becoming a major methodology to build an enterprise system on it, because the developer can build the system without procuring and configuring hardware/software; thus, the customer can significantly reduce cost and time for development. The number of PaaS provider is expected to increase as the platform technology is becoming standardized or open-source software is available [36]–[44].

We assume that the price of PaaS is set on a per-process-per-hour basis for each type of service in this paper. This assumption is reasonable to simulate the existing PaaS model. For example, Heroku [45] charges for dynos, workers, databases and add-ons separately. The dyno is a front-end process responsible to HTTP requests ("more dynos provide more concurrency") and the worker is a back-end process responsible for queued jobs ("more workers provide more capacity"), both of which costs $0.05 per process per hour; while the database costs monthly depending on its performance.

3.2 Enterprise System

An enterprise system generally consists of multiple subsystems running in parallel and/or sequentially, each of which requires a guaranteed quality of service (QoS) at a predictable price. We assume that each enterprise system, or each user's request, is represented by workflow. An example of business workflow is a payroll system [46]. It consists of a payroll calculation task on Java service along with an employee database task on SQL service, followed by reporting task on PDF/Email service as shown in Fig. 1. Another example of engineering workflow is a CAE[†] system [47]. It consists of a mesh generation task and a CFD[††] analysis task on a HPC[†††] service, controlled by an optimization task on a general-purpose optimization service.

Every task needs to reserve the specified type of service within an appropriate timeslots to meet a deadline. The overall cost should also be restricted by the user's total budget. Each task in the workflow is implemented using PaaS; thus, the user needs combination of PaaS services to organize the user's workflow.

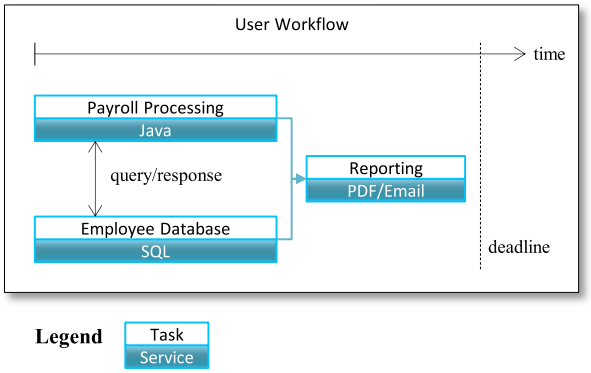

Fig. 1 Example enterprise application.

3.3 Marketplace

The service allocation decided by the marketplace must be fair and efficient; otherwise the user will have no incentive to take part in. Hence, we assume that the PaaS marketplace has to support the following requirements:

1. Combination for a workflow: Each user needs to bundle multiple services with different start/finish times as mentioned above. The cloud marketplace should allow users to express complementary requirements for an arbitrary combination of services.
2. Predictability and flexibility: Since supply and demand in the cloud computing environment changes dynamically over time, users may desire predictable allocation in advance and adjustment at runtime.
3. Economic efficiency: Every user and provider desires a fair and efficient allocation of services. The cloud marketplace should maximize the benefit of the participants and should not waste any resource. To this end, it is preferable for the marketplace to adopt an exact optimization approach rather than a heuristic approach.
4. Double-sided competition: To encourage a fair exchange between providers and users, the prices should only depend on supply-demand condition, giving no structural advantage on a seller's (provider's) side or a buyer's (user's) side. The cloud marketplace should be designed after the double-sided auction model, meaning that the providers and the users compete with each other.

4. Market Mechanism

In this section we present the proposed market mechanism

[†]CAE: Computer Aided Engineering.
[††]CFD: Computational Fluid Dynamics.
[†††]HPC: High Performance Computing.

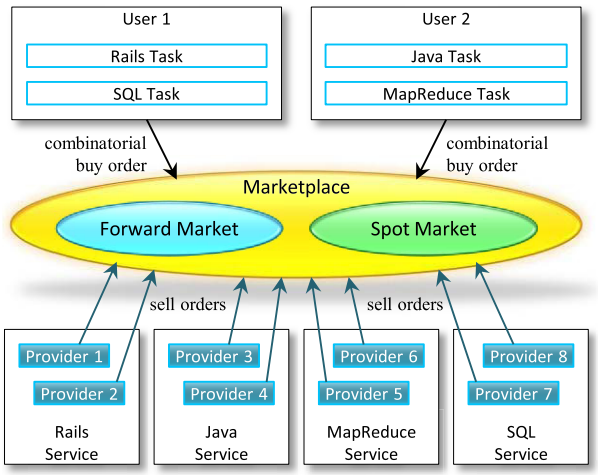

Fig. 2 Overview of the proposed marketplace.

in detail.

4.1 Overview

Figure 2 illustrates an overall perspective of the cloud computing environment with the proposed marketplace mechanism. The marketplace has two independent markets: the spot market for a short-term reservation (e.g. in one hour) and the forward market for a long-term reservation (e.g. in one month). The service providers participate in the markets as sellers while the users participate as buyers. For instance, the provider places sell orders with a market when he has a capacity of service with specific QoS. The user places a buy order with a market when he builds an application using specific services. The market accepts sell/buy orders for fixed duration, and then it determines the allocation of the services between the providers and the users. Finally, the market informs the participants of the resulting allocation to allow the user to deploy his application and the provider to preserve his capacity. We refer to the time of actual usage/provision as "delivery" and the sequence of the above procedures as a "round". The marketplace repeats the rounds periodically.

4.2 Service and Application

We assume that a service and an application satisfy the following conditions:

- An amount of resources that satisfy QoS is represented as a one-dimensional value, e.g. the number of processes or the performance of virtual machine. We refer to the amount of resources as "quantity" and represent it with the metric "units" in the rest of this paper. We assume that the price of resources is proportional to its quantity.
- A provider can host multiple users at the same time unless exceeding its capacity. For instance, the provider can provide 20 units to a user and 40 units for another user when it has a capacity of 60 units.
- A user can build an application using services offered by multiple providers to fulfill his demand. For instance, the user uses 10 units on the provider 1 and 30 units on the provider 2 when he builds an application consuming 40 units.
- An application can be migrated at runtime, i.e. an application running on a provider can be suspended, moved and resumed on another provider.

We omit in our model the physical parameters, such as network bandwidth and migration time, for the sake of simplicity. The physical cost can be included in a price or traded as a separate service in reality.

4.3 Trading Schedule

Each of the two markets holds clearinghouse auctions periodically. Figure 3 shows the schedule of the auctions. The spot and the forward auctions have the same procedure in different timescale. Here sellers and buyers are treated equally as participants.

In the spot market, a participant willing to sell/buy services in one-hour timeslot t needs to submit a request during the timeslot $t-2$, which begins two hours prior and ends one hour prior to the requested timeslot. For instance, let us suppose trading services at the third timeslot (from 2:00 to 3:00). The market opens for one hour at the first timeslot (from 0:00 to 1:00) to accept sell/buy orders from the participants. The market then closes and starts matchmaking, i.e. computes the optimal allocation of services from providers to users. Within one hour (by 2:00) the matchmaking finishes and the market notifies the results to the participants. Finally the participants utilize the services they win during the third timeslot (from 2:00 to 3:00).

Trading in the forward market is same as the spot market except the timescale of the procedure. Let F denotes the length of the forward delivery days[†] indicated by the blue-colored boxes marked 'forwards' in Fig. 3. A participant willing to sell/buy services starting on the f-th day needs to submit a request during the days between $f-(F-1)$ and $f-2$, that begins $F-1$ days prior and ends two days prior to the delivery day. For instance, the market opens a forward auction for one day (from 0:00 to 24:00) on the 1st day. Assuming $F=7$ the participants can place orders for services delivered between 0:00 on the 3rd day and 24:00 on the 9th day. The forward market performs matchmaking and notifies the results advising when to utilize the services. Note that the forward matchmaking can spend at most 22 hours; this is the design to support time-consuming MIP technique. Since the spot market opens at 22:00 for the next day's services, the forward matchmaking must finish before 22:00.

4.4 Bidding Language

The bidding language determines the information included

[†](financial term) The day on which seller/buyer actually provide/use the services.

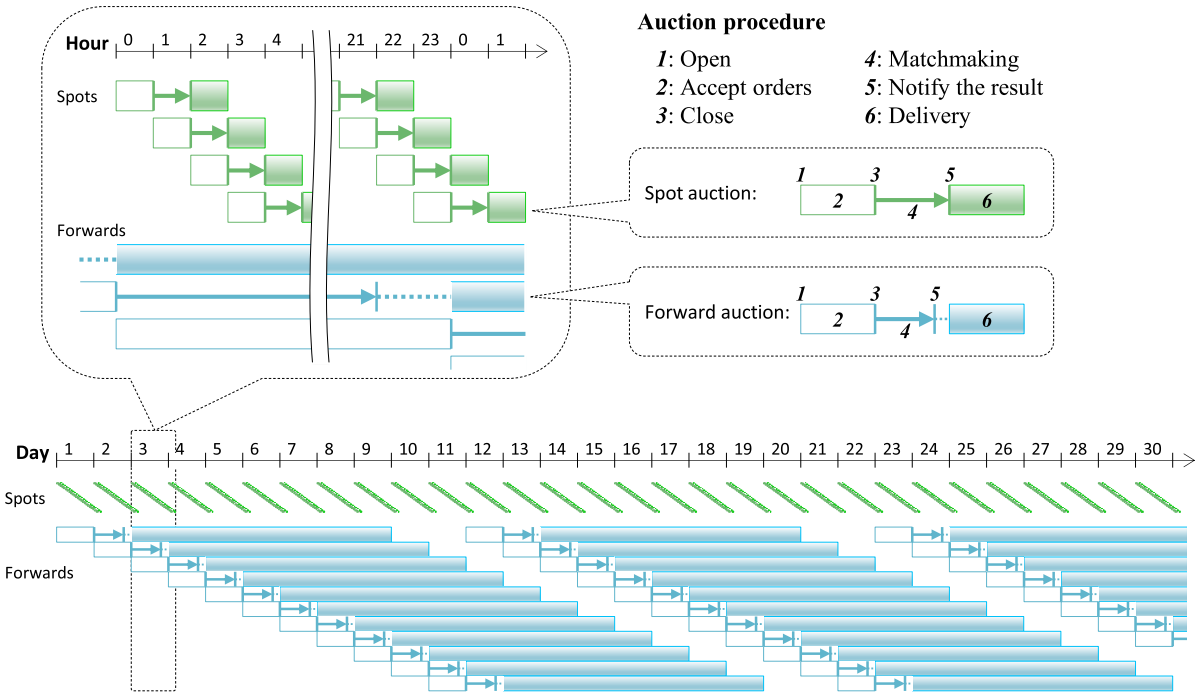

Fig. 3 Trading schedule.

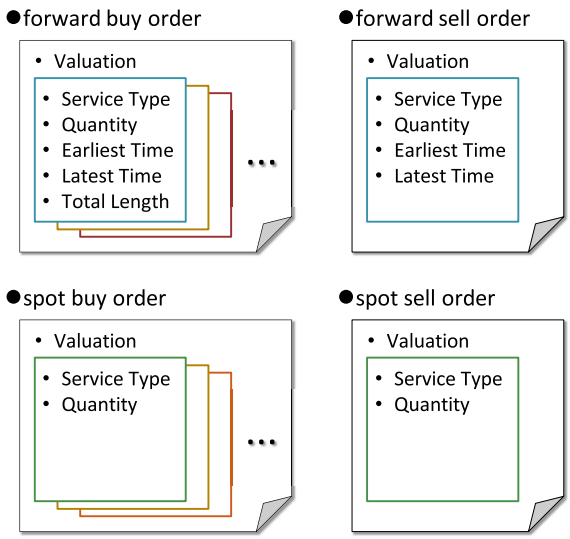

Fig. 4 Bidding language.

in an order. Figure 4 illustrates the order forms sent from a participant to a market.

A buy order from a user has a valuation (maximum price he wishes to pay) and a bundle of arbitrary services he needs. For each service the user specifies a service type, quantity, the earliest timeslot acceptable to start (arrival time of the task), the latest timeslot acceptable to finish (deadline of the task) and the total number of timeslots (estimated runtime of the task). The latter three parameters are only used in the forward market. Note that the valuation is given to a bundle of services, not to each discrete service, so that the user can express requirements for receiving multiple services in combination. If the market cannot reserve all the services in a bundle at once, the user receives nothing at all.

A sell order from a provider has a valuation (minimum price he wishes to earn) and a service he offers. The provider specifies a service type, quantity, the earliest timeslot and the latest timeslot available for use. The latter two parameters are only used in the forward market. Note that a sell order includes only one service. The provider can make separate orders for different services. If a provider wishes to sell certain low-level services at the same time, he can do so by bundling them into a single high-level service.

Formulation: Let $M = \{m_1, \ldots, m_{|M|}\}$, $m_i = \{v_i, S_i\}$ be sell orders; $N = \{n_1, \ldots, n_{|N|}\}$, $n_j = \{v_j, S_j\}$ be buy orders; and $G = \{g_1, \ldots, g_{|G|}\}$ be service types; $1 \le t \le T$ be timeslots; and v_i and v_j be valuation. A buy order is formulated as

$$O_j = \{(g_k, q_{j,k}, b_{j,k}, d_{j,k}, l_{j,k}) \mid 1 \le k \le |G|\}$$

where $q_{j,k}$ is the quantity of service g_k, $b_{j,k}$ is the earliest beginning timeslot, $d_{j,k}$ is the latest ending timeslot and $l_{j,k}$ is the number of timeslots[†]. Similarly, a sell order is formulated as

$$O_i = (g_k, q_{i,k}, b_{i,k}, d_{i,k}) \quad 1 \le k \le |G|.$$

4.5 Allocation Scheme

The allocation scheme determines the winners of an auction, or allocation of services from providers to users. Our

[†]If $l_j < d_j - b_j + 1$ then the task may be suspended/resumed during runtime.

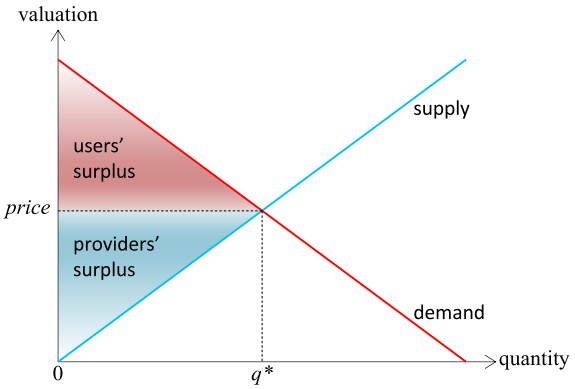

Fig. 5 Illustration of surplus and welfare (social economic welfare = users' surplus + providers' surplus).

goal is to get an economically efficient (or Pareto optimal) allocation of resources, where it is impossible to increase a participant's surplus without decreasing another participant's surplus. Here the surplus means the difference between the market price and the participant's internal valuation; i.e. (price − cost) for the provider and (utility − price) for the user, as shown in Fig. 5. The aggregate surplus, i.e. the difference between the buyers' valuation and the sellers' valuation, is also known as the social economic welfare.

Maximizing the social economic welfare w is the sufficient condition for Pareto optimality [48]. Therefore we formulate the winner determination problem into a linear mixed integer program (MIP) and try to exactly maximize w. Here, we introduce four decision variables: $u_j \in \{0,1\}$ denotes whether the buyer n_j gets all services in the bundle; $x_{j,k} \in \{0,1\}$ denotes whether the service g_k is allocated to the buyer n_j; $z_{j,k,t} \in \{0,1\}$ denotes whether the service g_k is allocated to the buyer n_j in the timeslot t; $0 \le y_{i,j,k,t} \le 1$ denotes the percentage of the service allocated to the buyer n_j in the timeslot t, where the service g_k is owned by the seller m_i. The solver then maximizes the social economic welfare w by solving the MIP:

Maximize

$$w = \sum_{j=1}^{|N|} v_j u_j - \sum_{i=1}^{|M|}\sum_{j=1}^{|N|}\sum_{k=1}^{|G|}\sum_{t=1}^{T} v_i y_{i,j,k,t} \tag{1}$$

s.t.

$$\sum_{k=1}^{|G|} x_{j,k} - |G| u_j = 0, \quad 1 \le j \le |N| \tag{2}$$

$$\sum_{t=1}^{T} z_{j,k,t} - l_{j,k} x_{j,k} = 0, \quad 1 \le j \le |N|,\ 1 \le k \le |G| \tag{3}$$

$$\sum_{j=1}^{|N|} y_{i,j,k,t} \le 1, \quad = 1 \le i \le |M|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{4}$$

$$q_{j,k} z_{j,k,t} - \sum_{i=1}^{|M|} q_{i,k} y_{i,j,k,t} = 0, \quad 1 \le j \le |N|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{5}$$

$$(b_{j,k} - t) z_{j,k,t} \le 0, \quad 1 \le j \le |N|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{6}$$

$$(t - d_{j,k}) z_{j,k,t} \le 0, \quad 1 \le j \le |N|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{7}$$

$$(b_{i,k} - t) \sum_{j=1}^{|N|} y_{i,j,k,t} \le 0, \quad = 1 \le i \le |M|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{8}$$

$$(t - d_{i,k}) \sum_{j=1}^{|N|} y_{i,j,k,t} \le 0, \quad 1 \le i \le |M|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{9}$$

$$u_j \in \{0,1\}, \quad 1 \le j \le |N| \tag{10}$$

$$x_{j,k} \in \{0,1\}, \quad 1 \le j \le |N|,\ 1 \le k \le |G| \tag{11}$$

$$z_{j,k,t} \in \{0,1\}, \quad 1 \le j \le |N|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{12}$$

$$0 \le y_{i,j,k,t} \le 1, \quad 1 \le i \le |M|,\ 1 \le j \le |N|,\ 1 \le k \le |G|,\ 1 \le t \le T \tag{13}$$

4.6 Pricing Scheme

A price earned/paid by a provider/user for an allocation is decided by the pricing scheme. The pricing scheme should be budget balanced and individually rational in order to sustain the market and give providers/users incentives to participate in the market. The former means that total earnings of providers should equal the total payment of users, and the latter means a provider/user earns/pays no less/more than their valuation.

We employ the K-pricing scheme [49] to meet the above requirements. The basic idea of K-pricing is to distribute the social economic welfare among the users and the providers. It is straightforward in non-combinatorial auctions. In our combinational auctions, however, we can neither calculate the discrete price for each service nor for each timeslot of a user's order. Here, we propose the following algorithm to determine the price.

We assume $u_j = 1$, since the only orders that succeed need pricing. Let $0 \le K \le 1$ be an arbitrary fraction. For a buy order n_j, let w_j be the welfare corresponding to n_j, p_j be the price, $p_{i,j}$ be the price earned by the provider i, and $r_{i,j,k,t}$ be the proportion of the provider i's valuation to all the providers' valuation of the service k in timeslot t. They are formulated as

$$w_j = v_j - \sum_{i=1}^{|M|}\sum_{k=1}^{|G|}\sum_{t=1}^{T} v_i y_{i,j,k,t}, \tag{14}$$

$$p_j = v_j - (1 - K)w_j, \tag{15}$$

$$r_{i,j,k,t} = \frac{v_i y_{i,j,k,t}}{\sum_{i=1}^{|M|}\sum_{k=1}^{|G|}\sum_{t=1}^{T} v_i y_{i,j,k,t}}, \tag{16}$$

$$p_{i,j} = \sum_{k=1}^{|G|}\sum_{t=1}^{T} v_i y_{i,j,k,t} + K\sum_{k=1}^{|G|}\sum_{t=1}^{T} w_j r_{i,j,k,t}. \tag{17}$$

Consequently, the provider i's total earning p_i is

$$p_i = \sum_{j=1}^{|N|}\sum_{k=1}^{|G|}\sum_{t=1}^{T} v_i y_{i,j,k,t} + K\sum_{j=1}^{|N|}\sum_{k=1}^{|G|}\sum_{t=1}^{T} w_j r_{i,j,k,t}. \tag{18}$$

The incentive compatibility, which means that the participant's dominant strategy is to reveal his valuation truthfully, is another important aspect of the pricing scheme. However, these three aspects —the budget balance, the individual rationality and the incentive compatibility— cannot be fulfilled at the same time [50]. In W-Mart, we focus on the first two aspects, the budget balance and the individual rationality, because we consider non-truthful bidding should also be allowed as the participant's strategy.

5. Simulator

We developed a simulator, named W-Mart, to explore market behavior by means of multi-agent simulations. The overall architecture of W-Mart is shown in Fig. 6.

The W-Mart server and the machine agents are implemented as a Java class. Two markets with matchmaking/pricing mechanism are also implemented as Java class, running their own threads being synchronized by the server. The machine agents have their own demand and strategies to make orders. They can be run on separate machine or on the same machine, talking a dedicated text-based protocol over TCP/IP to communicate with the server. The human agents can also take place using same protocol, trading with the machine agents at the same time†. The winner determination problem is solved by an external general-purpose solver, which can be CPLEX [51] or lp_solve [52]. The overall architecture is designed after U-Mart [35] and the auction mechanism is built using MACE framework [53].

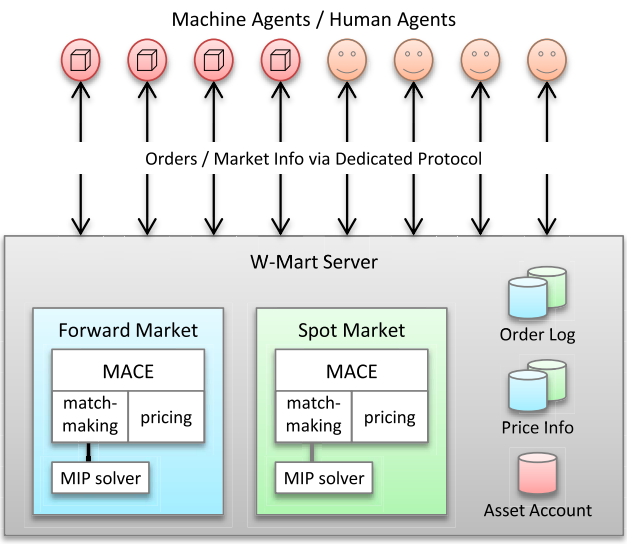

Fig. 6 Overview of W-Mart simulator.

6. Evaluation

We conducted three experiments to study the performance of the proposed mechanism. Figure 7 illustrates who takes part in the experiments.

First, in the single-market experiment, we ran one of two market mechanisms (the forward market and the spot market) to see the difference between them. In addition, we employed one of two kinds of buyer agent (the combinatorial buyer who makes a combined order and the separate buyer who makes separate orders for each service) to see the effectiveness of combinatorial auctions. Consequently, we compared the performance of four scenarios: (1) *forward/combinatorial*, (2) *forward/separate*, (3) *spot/combinatorial* and (4) *spot/separate*.

Second, in the dual-market experiment, we ran the forward market and the spot market simultaneously to verify the independency between them.

6.1 Simulation Settings

The simulations are carried out with the settings described below. Table 1 summarizes all the parameters used in the simulations.

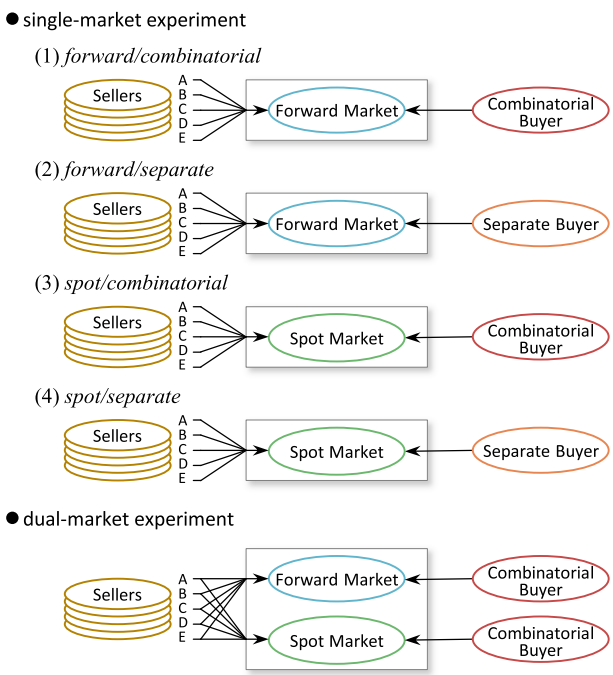

Fig. 7 Actorsof experiments.

†Human agent was not used for this paper.

Table 1 Summary of simulation parameters.

			Single market experiment	Dual market experiment
Market	Duration of operation		$T = 720$ hours	
	Forward delivery days		$F = 7$ days	
Seller	Service type		$G = \{A, B, C, D, E\}$	
	Quantity of a service		$q_i = 100$ units	
	Order price (per unit per hour)		$\frac{v_i}{q_i(d_i - b_i + 1)} = 1$ cent	
Buyer	Service type		$G = \{A, B, C, D, E\}$	
	Number of task in a workflow		$H \in \{2, 3, 4, 5\}$	
	Quantity of service required by a task		$q_j = [1, 100]$ units, uniform dist.	
	Order price (per unit per hour)		$\frac{v_j}{q_j l_j} = [2, 10]$ cents, uniform dist.	
	Length of a workflow		$l_j = [2, 24]$ hours, exponential dist. ($\lambda = 0.25$) $l_j = d_j - b_j + 1$	
	Length of a task	Leading tasks	$l_{j,1}, \ldots, l_{j,H-1} = [1, l_j - 1]$ hours, uniform dist.	
		Following task	$l_{j,H} = l_j - l_{j,1}$ hours	
	Margin	Forward	$[2, 7]$ days, uniform dist.	
		Spot	2 hours	
	Arrival rate (Poisson Arrival, $k = 2\lambda$)	Forward	$\lambda = \{1, 3, 5, 7, 10, 14, 20, 30, 40, 50\}$	$\lambda = 10$
		Spot		$\lambda = \{1, 3, 5, 7, 10, 14, 20, 30\}$

A) Market

One timeslot is one hour and one day is 24 timeslots. The market operates for 30 days (720 hours). The forward market deals with seven days (168 hours) of future services and clears a round at every midnight. The spot market deals with one hour of services and clears a round every hour. This setting exactly implements the trading schedule described in Sect. 4.

B) Sellers

There are five sellers with different type of services, namely service A, B, C, D and E. A seller has an ability to provide constant units of his service every hour (the quantity is shown in Table 1). The order price is fixed to one cent per unit per hour, for the sake of simplicity. A seller agent attempts to sell all amount of his service as early as possible. For instance, after he ordered 100 units and contracted to sell 30 units in one round, he will order to sell remaining 70 units in the next round.

C) Buyers

All buyers have their own applications presented by workflows. Each workflow consists of two phases of tasks, namely the leading task(s) and the following task, as shown in Fig. 8. Each task requires service A, B, C, D or E, exclusively. The overall length of a workflow follows the exponential distribution with $\lambda = 0.25$, where the minimum length is two hours and the maximum is 24 hours. This means that 50% of the workflows have four hours or shorter

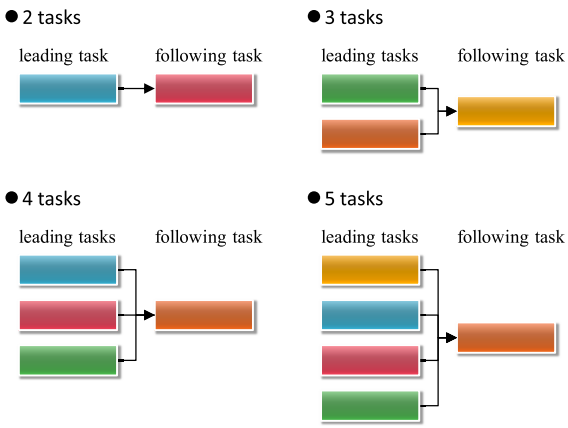

Fig. 8 Shapes of wofkflow.

length. The lengths of the leading tasks are set randomly within the overall length of the workflow and the following task spends the remainder. The valuation of a task follows the uniform distribution between 2 and 10 cents per unit per hour. The order price is the sum of the valuations of all tasks in the workflow.

The user places an order for a workflow before the time he actually start to use them. The margin of time between ordering and starting follows the uniform distribution between two and seven days for the forward orders, and is fixed to two hours for the spot orders. The quantity of each service follows the uniform distribution between 1 and 100 units.

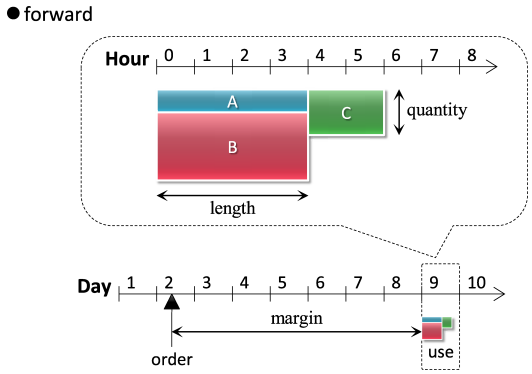

Fig. 9 Example forward buyer order.

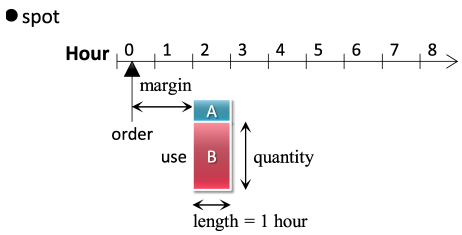

Fig. 10 Example spot buyer order.

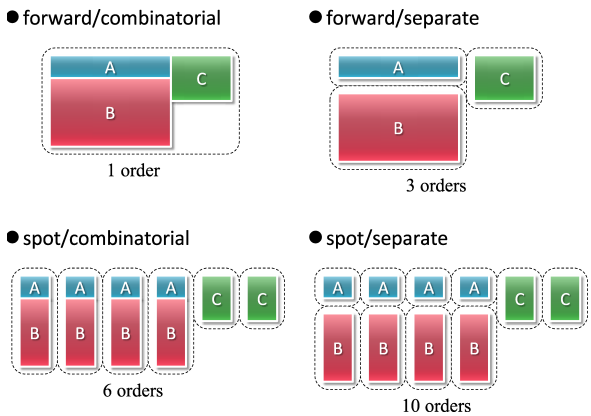

Fig. 11 Buyer order division.

Figure 9 and Fig. 10 show examples of orders placed with the forward market and with the spot market, respectively. We assume that arrival of workflows follows the Poisson Arrival with parameters shown in Table 1. We conducted 100 times of simulation run for each arrival rate and show the average results.

A buyer agent attempts to buy all services in a workflow. In *forward/combinatorial* scenario he orders all services at once as a bundle; in other scenarios he divides the bundle into a set of orders in an appropriate manner. Figure 11 illustrates how to divide a workflow depending on the scenario. Here, a user needs to run a workflow of three tasks, for example, starting with four hours of services A and B followed by two hours of service C. In the *forward/combinatorial* scenario he puts one order for all services on the forward market at once. In the *forward/separate* scenario he puts three orders for each service on the forward market at once, i.e. each order is processed independently. In the *spot/combinatorial* scenario he puts six orders for each timeslot on the spot market for six times. In the *spot/separate* scenario he puts 10 orders for each service and timeslot on the spot market for six times. The workflow is fulfilled if he succeed to reserve all services required by the tasks in the workflow; otherwise the workflow is not fulfilled and the reserved services are wasted, i.e. paid but not utilized. In other words, the wasted services give the buyer no benefit while consuming his budget.

6.2 Performance Metrics

We used the following metrics to evaluate the performance of the market mechanisms.

The demand/supply ratio (D/S) indicates a load on the overall system. D/S is computed by (19).

$$\mathrm{D/S} = \frac{\sum_{j=1}^{|N|}\sum_{k=1}^{|G|} q_{j,k}}{\sum_{i=1}^{|M|}\sum_{k=1}^{|G|} q_{i,k}} \tag{19}$$

The workflow completion rate (WC) indicates the rate of the number of workflows fulfill their requirements compared to the total number of workflows. A higher rate means better performance. WC is computed by (20).

$$\mathrm{WC} = \frac{\sum_{j=1}^{|N|} u_j}{|N|} \tag{20}$$

The cost performance (CP) indicates the users' total valuation fulfilled (excluding wasted services) compared to the total payments (including wasted services) by the users. A higher value means better performance. CP is computed by (21).

$$\mathrm{CP} = \frac{\sum_{j=1}^{|N|} v_j u_j}{\sum_{j=1}^{|N|} p_j} \tag{21}$$

The global utilization (GU) indicates total quantity of services utilized in the market (i.e. reserved and not wasted) compared to the total quantity of services offered by providers. A higher utilization means better performance. GU is computed by (22).

$$\mathrm{GU} = \frac{\sum_{j=1}^{|N|}\sum_{k=1}^{|G|} q_{j,k} u_j}{\sum_{i=1}^{|M|}\sum_{k=1}^{|G|} q_{i,k}} \tag{22}$$

The market price (MP) indicates an average price per unit per hour decided in the market. The users/providers actually pays/earns this amount of money. A lower price means better for users. MP is computed by (23).

$$\mathrm{MP} = \frac{\sum_{i=1}^{|M|} p_i}{\sum_{i=1}^{|M|}\sum_{j=1}^{|N|}\sum_{k=1}^{|G|}\sum_{t=1}^{T} y_{i,j,k,t} q_{i,k}} \tag{23}$$

Note that the minimum MP in the experiments is 3.5. The reason is that an order price is one cent for a selling order and two to 10 cents in uniform distribution (six cents on average) for a buying order; thus, the K-pricing scheme calculates $(1 + 6)/2 = 3.5$.

6.3 Results

A) Single-market experiment

First we show the results of the single-market experiments to compare performance of four market mechanisms: *forward/combinatorial (fwd/cmb)*, *forward/separate (fwd/sep)*, *spot/combinatorial (spt/cmb)* and *spot/separate (spt/sep)*.

Figure 12 shows the workflow completion rate (WC). First we can see the advantage of the forward market mechanism. Users have more opportunity to buy services that satisfy the users' requests in the forward market. Second, the combinatorial market mechanism shows more advantage compared to the separate market mechanism. In the cloud computing model discussed in this paper, users need to reserve services for all tasks in the users' workflow. The combinatorial market releases all services for tasks in the workflow if it fails to fulfill the requirements, while the separate market keeps them. Thus, the separate market significantly waste services and degrade the performance. We conclude that *fwd/cmb* is the best mechanism to improve WC.

Figure 13 shows the cost performance (CP). CP indicates the effectiveness of the market mechanism from the users' point of view. The result shows that CP keeps the ideal value in *fwd/cmb* mechanism, because it guarantees the users to profit from all the services they pay for. In other mechanisms, in contrast, CP decreases monotonically because they cannot guarantee the users to complete their workflows while keeping the payments for the fragmented services.

Figure 14 shows the global utilization (GU). GU indicates the effectiveness of the market mechanism from the providers' point of view. Ideally speaking, GU can be equal to D/S where $D/S \leq 1$ and can be 1 where $D/S > 1$. The result shows that GU increases monotonically in *fwd/cmb* mechanism since it does not waste any services. In *fwd/sep* mechanism the similar trend is observed where $D/S \leq 1$. In the spot market mechanism, in contrast, GU is saturated quickly and begins to decrease because the excessive collision in the spot market makes most of the workflow incomplete and wastes significant amount of services.

Figure 15 shows the market price (MP). Again we see the advantage of the forward market mechanism, in which the users can buy services at significantly low prices com-

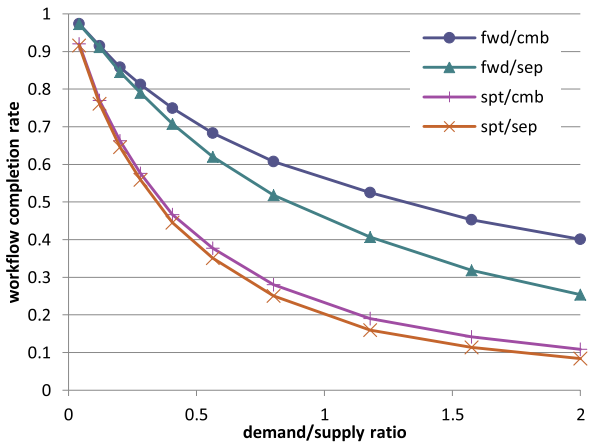

Fig. 12 Workflow completion rate (single market).

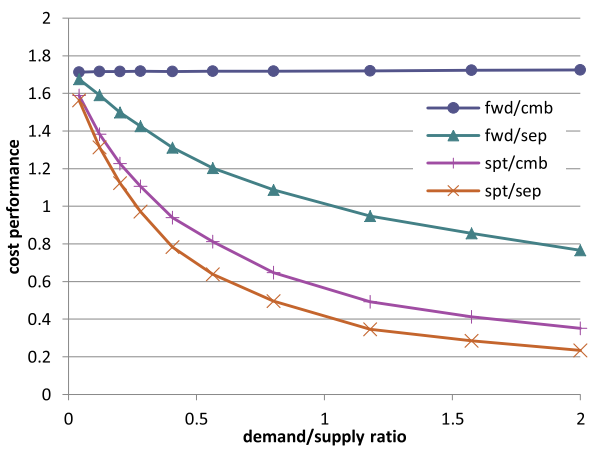

Fig. 13 Cost performance (single market).

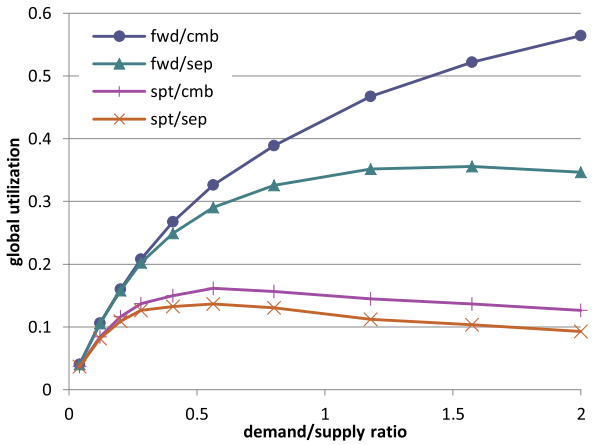

Fig. 14 Global utilization (single market).

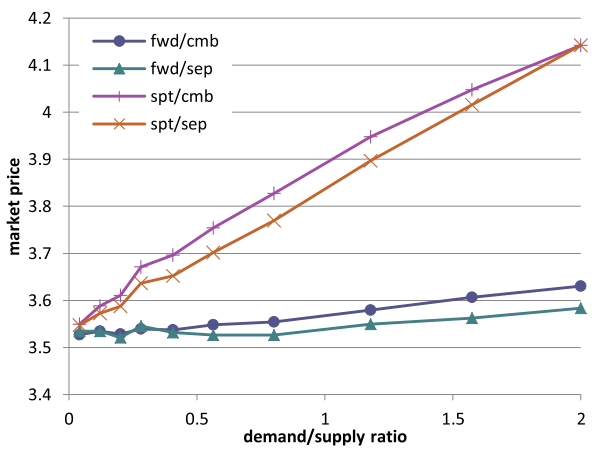

Fig. 15 Market price (single market).

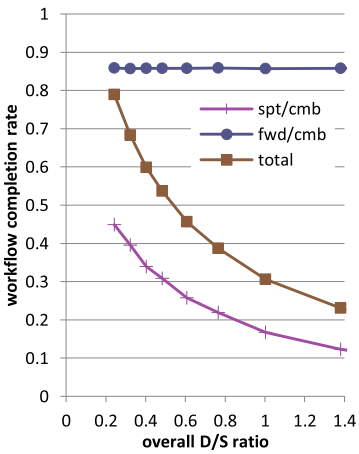

Fig. 16 Workflow completion rate (dual market, D/S of *fwd/cmb* = 0.2).

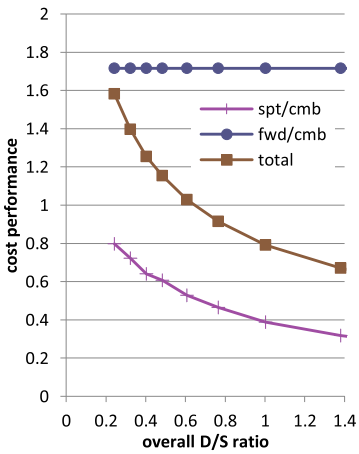

Fig. 17 Cost performance (dual market, D/S of *fwd/cmb* = 0.2).

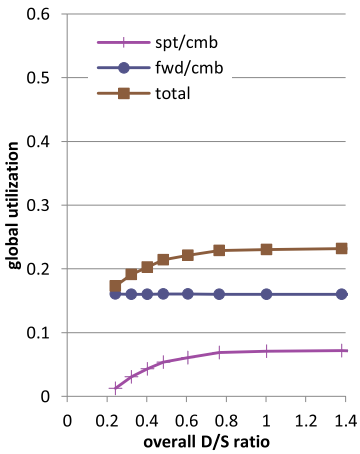

Fig. 18 Global utilization (dual market, D/S of *fwd/cmb* = 0.2).

pared to those of the spot market mechanism. Remember that the market mechanism determines winners among users' requests according to their valuation. Users undergo such competitions for every timeslots in the spot market mechanism, buying services at unreasonably high prices and wasting most of them. In the forward market mechanism, in contrast, users experience fewer competitions for entire reservations, resulting in reasonable prices and efficient utilization of services.

The performance of conventional scheduling systems is estimated to be similar with the separate market mechanism, since no conventional scheduler is able to reserve a combination of resources for a workflow. The simulation results demonstrate that *fwd/cmb* mechanism always outperforms the separate market mechanisms in WC and GU[†]. Consequently, we conclude that the proposed *fwd/cmb* mechanism outperforms any conventional scheduling systems under the condition discussed in this paper.

B) Dual-market experiment

Next we show the results of the dual-market experiment to verify the independency between the forward market and the spot market running simultaneously. In this scenario the forward contracts should have a priority over the spot contracts; otherwise the forward trading cannot serve as an advance reservation. We verified it by deploying two users: one takes part in *fwd/cmb* with a constant load (D/S = 0.2)[††] and the other takes part in *spt/cmb* while changing its load. The providers place the order first at the forward market to sell as much as possible and next at the spot market to sell the remainder.

Figure 16, Fig. 17 and Fig. 18 shows the workflow completion rate (WC), the cost performance (CP) and the global utilization (GU), respectively. Their x axes indicate the aggregate load of two markets. The plots indicate that the performance of *fwd/cmb* does not fluctuate and is not affected by the spot market. Thus the users can rely on the forward market to make an advance reservation. At the same time, the spot market also works well; it contributes to the providers utilizing the remainder resources and to the users procuring immediate resources.

7. Conclusion and Future Work

In this paper we proposed a combinatorial auction-based marketplace mechanism with exact optimization technique. The experiments compared four types of market design. The results showed that the *forward/combinatorial* design brings the best completion rate and cost performance for the users as well as the highest global utilization and second lowest market price. The results also showed that operating simultaneously the forward market and the spot market did not disturb the advance reservations while increased the global utilization.

However, the priority of the forward contracts over the spot contracts relies on the providers' faithful reservation policy. If the providers adopt an aggressive policy like double-booking, some of the forward contracts may be canceled at runtime, and thus it may not serve as an advance reservation. Some additional mechanism like margin[†††], penalty or reputation system may be needed to secure the predictability of resource allocation.

Sophisticated strategies of seller/buyer agents can significantly improve the performance of the market. For instance, a buyer agent can reduce wasted resources and can increase workflow completion rate by employing a smarter strategy to make his orders. Moreover, it is essential for seller/buyer agents to adjust their order price according to the market price in competition with each other[††††].

Our future work therefore includes investigation of

[†]CP and MP is not comparable with conventional schedulers.

[††]We also carried out simulations with other loads but omit their results since they have almost the same trend.

[†††](financial term) A deposit money to hedge the credit risk.

[††††]The first fundamental theorem of welfare economics states that a competitive equilibrium among participants leads to a Pareto efficient allocation of resources [54]. This is also known as the invisible hand of God.

market behavior using more sophisticated strategies of seller/buyer agents. We also plan to publish the source code of our simulator.

References

[1] "Microsoft Windows Azure" http://www.microsoft.com/windowsazure/
[2] "Google App Engine" http://code.google.com/appengine/
[3] I. Foster and C. Kesselman, "The grid: Blueprint for a new computing infrastructure," Oct. 1998.
[4] S. Clearwater, Market-Based Control: A Paradigm for Distributed Resource Allocation, World Scientific, 1996.
[5] J. Shneidman, C. Ng, D.C. Parkes, A. AuYoung, A.C. Snoeren, A. Vahdat, and B. Chun, "Why markets could (But don't currently) solve resource allocation problems in systems," Challenges, p.7, 2005.
[6] P. Cramton, Y. Shoham, and R. Steinberg, Combinatorial Auctions, The MIT Press, 2005.
[7] A. Das and D. Grosu, "Combinatorial auction-based protocols for resource allocation in grids," Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, 2005.
[8] Y. Shoham and K. Leyton-Brown, Multiagent systems: algorithmic, game-theoretic, and logical foundations, Cambridge University Press, 2009.
[9] S. de Vries and R.V. Vohra, "Combinatorial auctions: A survey," INFORMS J. Comput., vol.15, pp.284–309, July 2003.
[10] M.H. Rothkopf, A. Pekec, and R.M. Harstad, "Computationally Manageable Combinatorial Auctions," April 1995.
[11] Y. Fujishima, K. Leyton-Brown, and Y. Shoham, "Taming the computational complexity of combinatorial auctions," pp.548–553, 1999.
[12] T. Sandholm, "Algorithm for optimal winner determination in combinatorial auctions," Artif. Intell., vol.135, pp.1–54, Feb. 2002.
[13] A. Andersson, M. Tenhunen, and F. Ygge, Integer programming for combinatorial auction winner determination, Proc. 4th International Conference on MultiAgent Systems, pp.39–46, July 2000.
[14] P. He, Y. Li, Z. Nie, and N.E. Shawwa, Review of Linear Programming Software, 2007.
[15] "Linear programming," From Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Linear_programming
[16] I. Fujiwara, K. Aida, and I. Ono, "Applying double-sided combinational auctions to resource allocation in cloud computing," 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet, IEEE, pp.7–14, 2010.
[17] C.S. Yeo and R. Buyya, "A taxonomy of market-based resource management systems for utility-driven cluster computing," Software: Practice and Experience, vol.36, pp.1381–1419, Nov. 2006.
[18] R. Buyya, D. Abramson, and S. Venugopal, "The grid economy," Proc. IEEE, vol.93, pp.698–714, March 2005.
[19] J. Broberg, S. Venugopal, and R. Buyya, "Market-oriented grids and utility computing: The state-of-the-art and future directions," J. Grid Computing, vol.6, pp.255–276, Dec. 2007.
[20] R. Buyya, C.S. Yeo, and S. Venugopal, Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, IEEE, 2008.
[21] R. Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing," Concurrency and Computation: Practice and Experience, vol.14, pp.1175–1220, Nov. 2002.
[22] R. Buyya, R. Ranjan, and R.N. Calheiros, Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities, IEEE, 2009.
[23] C.A. Waldspurger, T. Hogg, B.A. Huberman, J.O. Kephart, and W.S. Stornetta, "Spawn: A distributed computational economy," IEEE Trans. Softw. Eng., vol.18, no.2, pp.103–117, Feb. 1992.
[24] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid," Computer, vol.1, pp.283–289, 2000.
[25] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," International Journal of Supercomputer Applications, vol.11, pp.115–128, 1997.
[26] A. AuYoung, B.N. Chun, A.C. Snoeren, and A. Vahdat, "Resource allocation in federated distributed computing infrastructures," Proc. 1st Workshop on Operating System and Architectural Support for the Ondemand IT Infrastructure, pp.1–10, 2004.
[27] Z. Tan and J.R. Gurd, "Market-based grid resource allocation using a stable continuous double auction," Grid Computing, 2007 8th IEEE/ACM International Conference on, pp.283–290, 2007.
[28] B. Schnizler, D. Neumann, D. Veit, and C. Weinhardt, "Trading grid services – A multi-attribute combinatorial approach," Eur. J. Oper. Res., vol.187, pp.943–961, June 2008.
[29] "PBS Works" http://www.pbsworks.com/
[30] W. Gentzsch, "Sun grid engine: Towards creating a compute power grid," Proc. first IEEEACM International Symposium on Cluster Computing and the Grid, IEEE Computer Society, pp.35–36, 2001.
[31] M. Litzkow, M. Livny, and M.W. Mutka, "Condor - A hunter of idle workstations," Proc. 8th International Conference of Distributed Computing Systems, IEEE, pp.104–111, 1988.
[32] "Amazon EC2 Spot Instances" http://aws.amazon.com/ec2/spot-instances/
[33] "Heroku | Add-ons" http://addons.heroku.com/
[34] K. Hoki, "Outline of Japan electric power exchange (JEPX)," Trans. Inst. Electr. Eng. Jpn. B, vol.125, pp.922–925, 2005.
[35] H. Kita, H. Sato, N. Mori, and I. Ono, "U-mart system, software for open experiments of artificial market," Computational Intelligence in Robotics and Automation 2003 Proceedings 2003 IEEE International Symposium on, Ieee, vol.3, pp.1328–1333, 2003.
[36] P.T. Endo, G.E. Gonçalves, J. Kelner, and D. Sadok, "A survey on open-source cloud computing solutions," Comput. Netw., pp.3–16, 2009.
[37] T.D. Cordeiro, D.B. Damalio, P.T. Endo, A.V. De Almeida Palhares, G.E. Gonçalves, D.F.H. Sadok, J. Kelner, B. Melander, V. Souza, and J.-E. Mångs, "Open source cloud computing platforms," 2010 Ninth International Conference on Grid and Cloud Computing, pp.366–371, 2010.
[38] "Cloud Foundry" http://cloudfoundry.com/
[39] "OpenCloud" http://www.opencloud.com/
[40] "Open Compute Project" http://opencompute.org/
[41] "OpenStack" http://openstack.org/
[42] "OpenShift" http://openshift.redhat.com/app/
[43] "RightScale" http://www.rightscale.com/
[44] "Engine Yard" http://www.engineyard.com/
[45] "Heroku" http://www.heroku.com/
[46] "Cloud Computing Use Cases White Paper Version 2.0" http://www.opencloudmanifesto.org/Cloud_Computing_Use_Cases_Whitepaper-2_0.pdf
[47] S. Weston, D. Green, G. Katsaros, T. Seed, N. Mc Donnell, and F. Scharinger, GridCAE Grid for Computer Aided Engineering, 2009.
[48] A. Mas-Colell, M.D. Whinston, and J.R. Green, "Microeconomic theory," The Canadian Journal of Economics, vol.21, p.436, 1995.
[49] M. Satterthwaite and S. Williams, "The Bayesian theory of the k-double auction," in The Double Auction Market: Institutions, Theories, And Evidence, ed. J. Rust, pp.99–123, Westview, 1993.
[50] R.B. Myerson and M. Satterthwaite, "Efficient mechanisms for bilateral trading," J. Economic Theory, vol.29, pp.265–281, 1983.
[51] "IBM ILOG CPLEX Optimizer" http://www.ibm.com/software/integration/optimization/cplex-optimizer/
[52] "lp_solve" http://tech.groups.yahoo.com/group/lp_solve/
[53] B. Schnizler, "MACE: A multi-attribute combinatorial exchange," Negotiation, Auctions, and Market Engineering, pp.84–100, 2008.
[54] A.M. Feldman and R. Serrano, Welfare economics and social choice theory, Springer, 2006.

Ikki Fujiwara received his B.S. and M.S. degrees in Engineering from Tokyo Institute of Technology in 2002 and 2004, respectively. He worked as a systems engineer from 2004 to 2008 at Hitachi, Ltd. to develop railroad operation systems. He is now a doctoral student at The Graduate University for Advanced Studies (SOKENDAI). He is a member of IPSJ and IEEE.

Kento Aida received Dr. Eng. in electrical engineering from Waseda University in 1997. He became a research associate at Waseda University in 1992. He joined Tokyo Institute of Technology and became a research scientist at the Department of Mathematical and Computing Sciences in 1997, an assistant professor at the Department of Computational Intelligence and Systems Science in 1999, and an associate professor at the Department of Information Processing in 2003, respectively. He is now a professor at National Institute of Informatics and a visiting professor at the Department of Information Processing in Tokyo Institute of Technology from 2007. He was also a researcher at PRESTO in Japan Science and Technology Agency (JST) from 2001 through 2005, and a research scholar at the Information and Computer Sciences Department in University of Hawai'i in 2007.

Isao Ono received his B.S. degree from the Department of Control Engineering, Tokyo Institute of Technology, Tokyo, Japan, in 1994. He received Dr. of Engineering at Tokyo Institute of Technology, Yokohama, in 1997. He worked as a Research Fellow from 1997 to 1998 at Tokyo Institute of Technology, and at University of Tokushima, Tokushima, Japan, in 1998. He worked as a Lecturer from 1998 to 2001 and an associate professor from 2001 to 2005 at University of Tokushima. He has been working as an associate professor at Tokyo Institute of Technology since 2005. His research interests include evolutionary computation, artificial intelligence and grid computing. He is a member of JSAI, SCI, and SICE.