2010 10th Annual International Symposium on Applications and the Internet

Applying Double-sided Combinational Auctions to Resource Allocation in Cloud Computing

Ikki Fujiwara
The Graduate University for Advanced Studies (SOKENDAI)
Tokyo, Japan
ikki@nii.ac.jp

Kento Aida
National Institute of Informatics/ Tokyo Institute of Technology
Tokyo, Japan
aida@nii.ac.jp

Isao Ono
Tokyo Institute of Technology
Kanagawa, Japan
isao@dis.titech.ac.jp

***Abstract*— We believe that a market-based resource allocation will be effective in a cloud computing environment where resources are virtualized and delivered to users as services. We propose such a market mechanism to allocate services to participants efficiently. The mechanism enables users (1) to order a combination of services for workflows and co-allocations and (2) to reserve future/current services in a forward/spot market. The evaluation shows that the mechanism works well in probable setting.**

Keywords- cloud computing, web services, market, auction, scheduling, optimization, linear programming

I. INTRODUCTION

Cloud computing is an emerging paradigm for distributed computing environments. The computing resources, either software or hardware, are virtualized and allocated as services from providers to users. QoS is an important issue for industrial users. Advanced features related to QoS include performance guarantee of service, co-allocation of multiple services, and support of a workflow involving different services.

We envision that providers in the near future will compete to offer cloud computing services and that

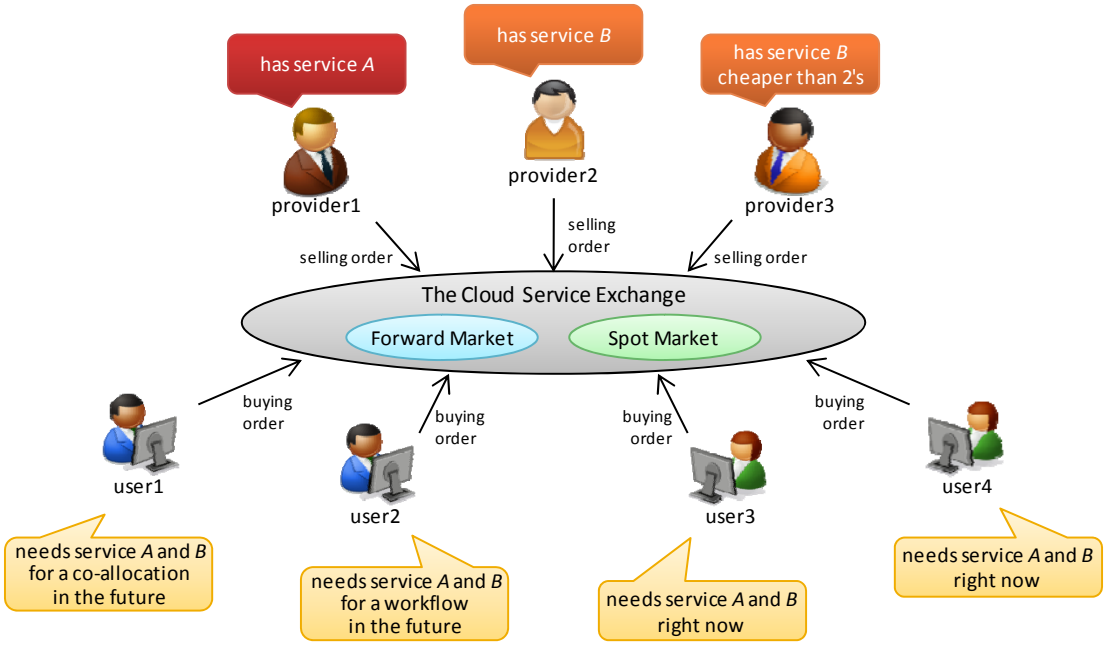

Figure 1. Overview of the Cloud Service Exchange

978-0-7695-4107-5/10 $26.00 © 2010 IEEE
DOI 10.1109/SAINT.2010.93

thousands of users will compete to receive such services to run complex tasks with guaranteed QoS and limited budgets. However, an efficient allocation mechanism among service providers and users has yet to be devised.

In this paper, we propose a market-based resource allocation mechanism that allows participants to trade their services by means of a double-sided combinational auction. A market mechanism may be better able to cope with situations where large numbers of participants trade different services. The proposed mechanism enables participants to trade future and current services in the forward market and the spot market, respectively.

The rest of the paper is organized as follows. Section II describes the model of the cloud computing environment and the related work. Section III presents the proposed market mechanism. Section IV describes our simulator for the proposed mechanism, and Section V discusses the preliminary evaluation. Section VI summarizes our contributions and outlines future work.

II. Model and Related Work

A. Cloud Computing Environment Model

We assume the cloud computing environment includes service providers, users, and a market. The service providers and the users sell/buy different computing resources abstracted as services through the market. The services vary in abstraction level, from primitive resources (e.g. CPU cycles, storage capacity, and network bandwidth) to sophisticated applications (e.g. customer behavior analysis, structural mechanics simulations, and pay-check calculations). The users get one or more services together to meet their business requirements. Their requirements vary in regularity from periodical, scheduled tasks to temporal, immediate tasks. For instance, a user may run a financial risk evaluation program using 80 CPUs and 16GB of storage for 4 hours after midnight on every day.

The resource allocation mechanism needs to satisfy the service providers and the users. We define four requirements, and our goal is to devise the market-based resource allocation mechanism that satisfies them.

1) Economic efficiency: When the allocation is economically efficient, it is impossible to increase a participant's welfare without decreasing another participant's welfare; i.e. there is no wasted resource. Maximizing the total welfare is a sufficient condition for economic efficiency. The proposed mechanism employs mixed integer programming to strictly maximize the total welfare.

2) Predictability and flexibility: Since supply and demand in cloud computing environment changes dynamically over time, users may desire a predictable allocation in advance and a adjustment at runtime. The proposed mechanism installs two markets to support them: a forward market for advance reservations and a spot market for immediate reservations.

3) Combination for a workflow: Many cloud computing applications utilize several services in a specific order to organize a workflow. The user therefore needs to bundle multiple services with different start/finish times. The proposed mechanism allow users to express complementary requirements for an arbitrary combination of services, in order to support both workflows and co-allocations.

4) Double-side competition: To encourage a fair exchange between resource providers (sellers) and users (buyers), the prices should only depend on the supply-demand conditions. The proposed mechanism is based on the double-sided auction model [2] to give no advantage on the seller's side or the buyer's side.

B. Related Work

Market-based resource allocation has been a hot topic in grid literature for a decade. Schnizler et al. [1] introduced the notion of using a double-sided combinational auction to allocate grid resources. However, in this scheme, resources are bundled by the resource providers, and users cannot combine arbitrary resources in different timeslots to compose a workflow. Tan et al. [2] proposed a stable continuous double auction (SCDA). This auction is not truly combinational; i.e., users need to bid on multiple auctions in order to receive multiple resources. Amar et al. [3] illustrated a comprehensive grid market model including a futures market and a centralized/decentralized spot market. However, they did not discuss a model of the futures market. To the best of our knowledge, no previous work has studied a resource allocation mechanism that satisfies the requirements presented in II.A.

Although a computing resource market has yet to be realized at the industrial level, electricity markets have been in practical operation for several years. For instance, Japan Electric Power Exchange (JPEX) started operations in 2005. According to ref. [4], 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 computing 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 [5] 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. We are developing our evaluation framework to be compatible with the U-Mart system so that human agents can participate in experiments.

III. MARKET MECHANISM

Figure 1 shows a cloud computing environment with the proposed mechanism, the Cloud Service Exchange. There is a centralized exchange including the forward market and the spot market, where service providers sell their services and users buy these services to execute their tasks. The participants interact with the spot market and the forward market independently. Each service is represented as a uniform order and traded independently in the market, regardless of its abstraction level. A broker is one who buys many kinds of primitive resources in the market, composes them into a sophisticated application, and sells that application in the market.

We assume that the services satisfy the following conditions:

- The quantity of a service can be measured in arbitrary units (e.g. 60 requests/second of service A). We shall use arbitrary "units" in the rest of this paper.
- A service can be divided into an arbitrary fraction (e.g. a resource of 60 units is divided into 20 units for user 1 and 40 units for user 2).
- A task can be divided into sub-tasks and executed on multiple services (e.g. a task of 40 units runs on a service of 10 units from provider 1 and that of 30 units from provider 2).
- A task can be migrated at runtime (e.g. a task running on a service from provider 1 can be suspended and resumed on that from provider 2).

We omit the physical parameters (e.g. network distance between co-allocated services) for the sake of simplicity. Obviously, these physical matters affect the cost and the runtime in reality. Such a physical cost can be included in the service price or traded as a separate service.

The proposed mechanism has three main properties: (1) the bidding language defines the protocol between the participants and markets, (2) the allocation scheme determines the assignment of services, and (3) the pricing scheme fixes prices at which the participants trade their services. Below, we present the details of the properties in the forward and spot markets.

A. Forward Market

The forward market deals with long-term advance reservations by means of the clearinghouse auction. It performs matchmaking between sell orders and buy orders periodically. A provider/user sells/buys a service by the timeslot, and the timeslot is traded in the market. For instance, a user buying a service with the timeslot of 1 pm – 2 pm utilizes the service from 1 pm to 2 pm. The market accepts orders from providers/users any time and performs matchmaking periodically, e.g. every 24 hours.

1) Bidding Language

The bidding language describes the information in the orders from participants to the market.

A buy order from a user includes the following information:

- Valuation: the maximum price at which the user wishes to buy the bundle of services
- A bundle of arbitrarily services, each of which includes:
  - Name: the kind of service
  - Quantity: the amount (throughput) of the service
  - Earliest Time: the earliest acceptable timeslot to start the task
  - Latest Time: the latest acceptable timeslot to finish the task
  - Runtime: the number of timeslots to be allocated between the earliest and the latest time[1]

Note that the valuation is given to a bundle of services, not to each discrete service. In this way, the user can express requirements for receiving multiple services, e.g. co-allocation or workflow. 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 includes the following information:

- Valuation: the minimum price per timeslot at which the provider wishes to sell the service
- Name: the kind of service
- Quantity: the amount (throughput) of the service
- Earliest Time: the timeslot to begin the service
- Latest Time: the timeslot to end the service

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

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

where $q_{j,k}$ is the quantity of service g_k, $a_{j,k}$ is the earliest time, $d_{j,k}$ is the latest time and $l_{j,k}$ is the runtime. Similarly, a sell order is formulated as

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

2) Allocation Scheme

The allocation scheme determines the winners of an auction, or allocation of services to users. We formulate the winner determination problem into a linear mixed integer program (MIP) and try to strictly optimize the allocation. 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 total welfare w by solving the MIP:

[1] If ($Runtime < Latest\ Time - Earliest\ Time$), a task will be suspended and resumed at runtime.

Maximize

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

s.t.

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

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

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

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

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

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

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

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

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

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

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

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

3) 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 indicates 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 [6] to meet the above requirements. The basic idea of K-pricing is to distribute the welfare, which is the difference between the user's valuation and the provider's valuation. It is straightforward in single-service auctions. In our multiple-service 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

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

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

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

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

Consequently, the provider i's total earning p_i is

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

The incentive compatibility, which means that the participant's dominant strategy is to reveal his/her 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 [7]. In this paper, 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.

B. Spot Market

The spot market deals with short-term allocations. It deals with resources in the immediate timeslot. The bidding language, the allocation scheme, and the pricing scheme are almost the same as those of the forward market except that they have only one timeslot.

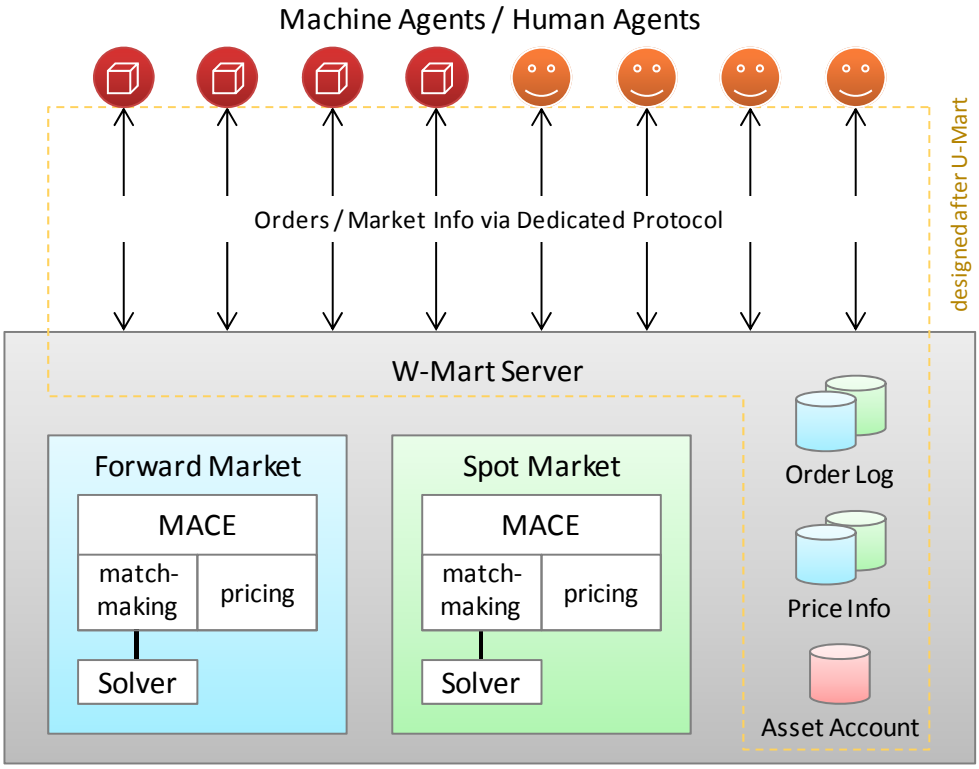

Figure 2. Simulator architecture

IV. SIMULATOR

We are developing a simulator system, named W-Mart, to explore market behavior by means of multi-agent simulations. The overall architecture of W-Mart is designed after U-Mart [5], as shown in figure 2.

U-Mart is a client-server system written in Java, including a dedicated text-based protocol over TCP/IP. The market server accepts sell/buy orders from the client agents, executes pricing and contracts, and manages the asset accounts. The client agents obtain the market information (e.g. current/historical prices) and make the orders depending on their own strategies. The agent can be either a machine or a human.

The original U-Mart system was specialized to trade a stock index futures of a given series of spot prices. It had a single instance of the futures market with hard-coded *itayose* algorithm [8] to perform matchmaking and pricing. Our W-Mart system, in contrast, has two market instances for the forward/spot markets run on separate threads, each of which is designed to trade arbitrary combinations of multiple goods. The protocol has been extended to support multiple markets and combinational orders. We have implemented the proposed matchmaking and pricing algorithm on top of MACE [1], which is a sophisticated framework for combinational auctions. This mechanism translates the orders into a mixed integer program (MIP) and solves it with a general-purpose LP/MIP solver, which can be CPLEX [9] or lp_solve [10].

V. EVALUATION

We carried out a preliminary evaluation to study the feasibility of the proposed mechanism. Currently no public marketplace for cloud services is in operation, and hence, no empirical guideline on evaluating a market mechanism has been established. Thus, we have two points of view in our preliminary evaluation: (1) verifying the combinational allocation by the market and (2) estimating the scalability of the market.

A. Verifing the Combinational Allocation

The proposed mechanism enables combinational allocations for workflows and co-allocations. In this section, we investigate two simple cases to see how combinational allocations are achieved in the forward market and in the spot market.

1) Experimental Settings

For the forward market, we assume four timeslots (i.e. from zero o'clock to four o'clock). Two kinds of services are offered by three providers: provider 1 offers service A; provider 2 and provider 3 offer service B with different prices. Two users require these services in different manners: user 1 needs services A and B simultaneously for co-allocation; user 2 needs services A and B sequentially for a workflow. The quantities, valuations and start/finish times of each service are shown in Figure 3. The required runtime of the task equals $(\text{finish time} - \text{start time})$, which means that no interruption occurs.

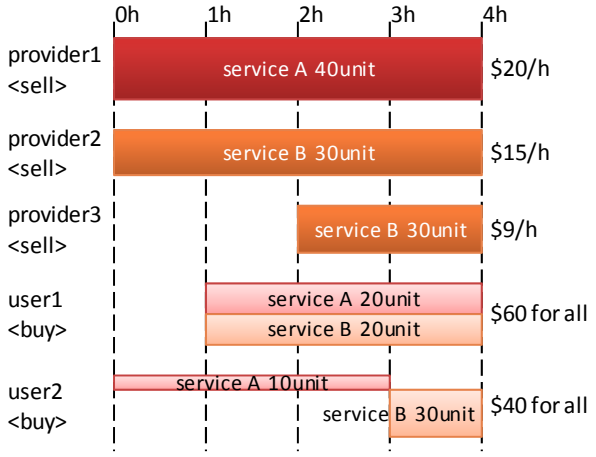

Figure 3. Combinational orders in the forward market

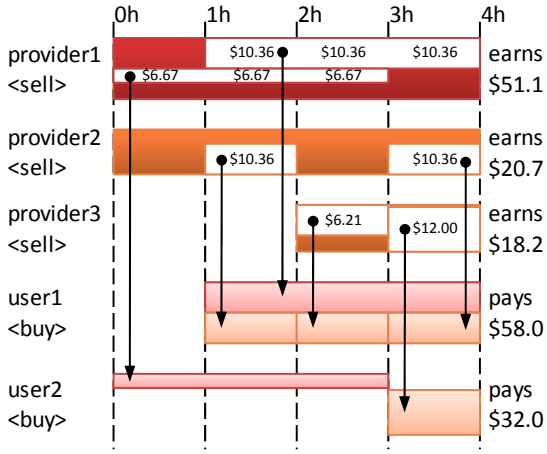

Figure 4. Combinational allocation in the forward market

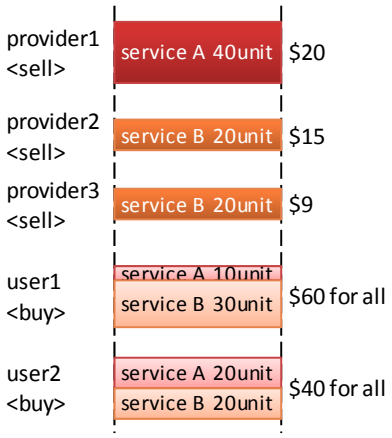

Figure 5. Combinational orders in the spot market

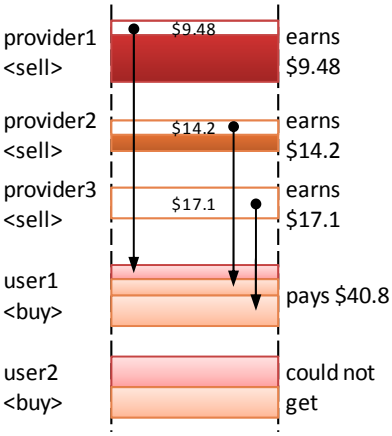

Figure 6. Combinational allocation in the spot market

Only one timeslot is available for the spot market. The providers and services are the same as those of the forward market. User 1 and user 2 require the same combination of services A and B, but the valuation of user 1 is higher than that of user 2.

Table I shows the formulations and table III shows the hardware and software configuration to run the simulator.

2) Results

Figure 3 and 4 respectively show the orders and the allocation results in the forward market. These results show that orders from all users are fulfilled. In particular, the order from user 2 consists of two tasks in a workflow, a task of service A and one of service B; and the services are properly allocated to the tasks. These results indicate that the proposed mechanism using the combinational auction properly allocated services to workflow tasks. The previous study was not able to do so [1]. Note that provider 3 won the competition to sell service B for user 1 in timeslot 2 because he priced it lower than provider 2 did and therefore generated more total welfare.

Figure 5 and 6 show orders and allocation results in the spot market. The supply of service B is less than the demand. As a result, user 2 lost the competition and bought nothing. Indeed provider 1 still has enough capacity for service A, but it is not allocated to user 2 since it does not fulfill the combinational order of user 2.

TABLE I. SIMULATION PARAMETERS (A)

Number of timeslots	$T = 4$	
Number of users	$\|N\| = 2$	
Number of providers	$\|M\| = 3$	
Number of services	$\|G\| = 2$	
Number of combined services	$nsrv = 2$	
Length of a task	$1 \leq len \leq 4$	$len = 1$ for spot market
Start time of a task	$0 \leq stt \leq T - 1$	$stt = 0$ for spot market

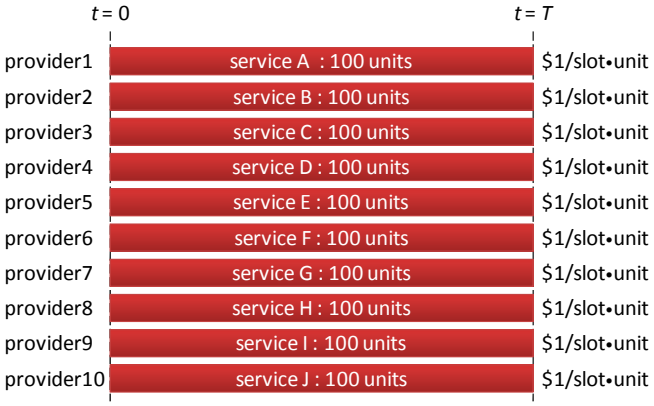

Figure 7. Sell orders in the simulation

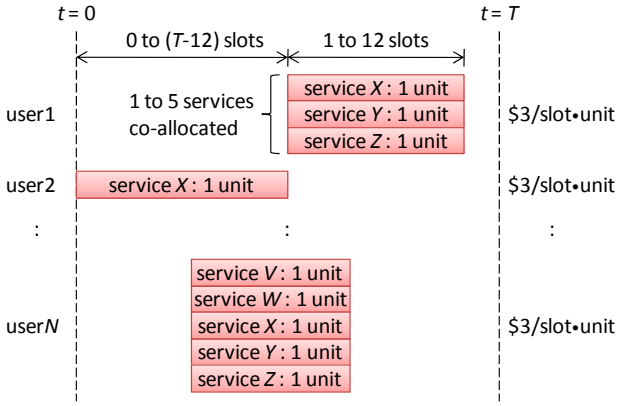

Figure 8. Buy orders in the simulation

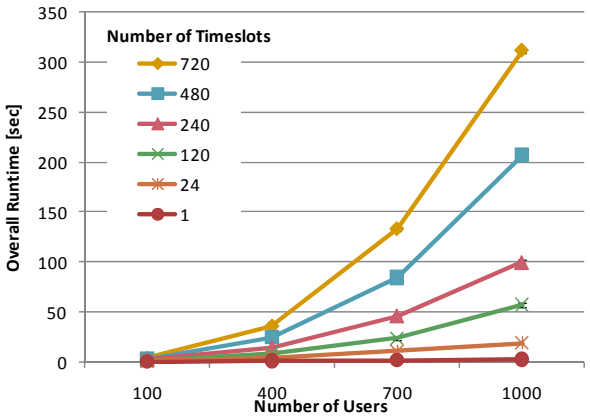

Figure 9. Overall runtime

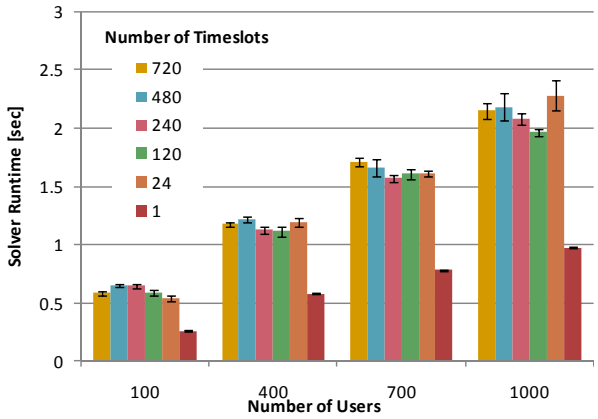

Figure 10. Solver runtime

B. *Estimating Scalability*

Mixed integer programming tends to consume a long time when faced with a large problem. In this section, we evaluate the scalability of the proposed mechanism in order to confirm its practicality in a cloud computing environment. The evaluation assesses the impact of the number of users and timeslots on the runtime.

1) *Experimental Settings*

We carried out the simulation by generating a set of orders and running the market mechanism. Since the evaluation aims to assess the scalability, we assume that the rounds are independent; i.e., the result of matchmaking of orders does not affect the next orders.

The number of timeslots has a range of {1, 24, 120, 240, 480, 720}. The case of #slots = 1 represents trading in the spot market, and other cases represent trading in the forward market. The actual time span covered by timeslots depends on the length of the timeslot. For example, #slots = 720 represents one month with a timeslot of one hour, or represents one year with a timeslot of 12 hours. We refer to the example of the Japanese electricity exchange for the time

TABLE II. SIMULATION PARAMETERS (B)

Number of timeslots	$T \in \{1, 24, 120, 240, 480, 720\}$	
Number of users	$\|N\| \in \{100, 400, 700, 1000\}$	
Number of providers	$\|M\| \in \{10\}$	
Number of services	$\|G\| \in \{10\}$	
Number of combined services	$1 \leq nsrv \leq 5$, uniform distribution	
Length of a task	$1 \leq len \leq 12$, uniform distribution	$len = 1$ for spot market
Start time of a task	$0 \leq stt \leq T - 12$, uniform distribution	$stt = 0$ for spot market
Selling quantity	100 units	
Buying quantity	1 unit for each service	
Seller's valuation	$1 per timeslot per unit	
Buyer's valuation	$3 per timeslot per unit	
Number of simulation runs	10 times	

TABLE III. SIMULATION ENVIRONMENT

CPU	AMD Opteron 8218 HE (2.6 GHz) 16 cores
RAM	32GB
OS	CentOS 5.1 (Linux kernel 2.6.18-92.el5)
JRE	Sun Java SE 1.6.0_11
Solver	ILOG CPLEX 11.200

granularity. We consider this extent of granularity to be applicable to the cloud computing environment.

The number of providers is set to 10, while the number of users has a range of {100, 400, 700, 1000}. Figure 7 shows the sell orders of the providers. Each provider offers a unique service and all the services are available anytime. Figure 8 shows the buy orders of the users. Each user requires one to five services chosen randomly out of 10 services to be co-allocated. The task length varies from one to 12 timeslots. The time margin between ordering and starting a task varies from zero to $(\#\text{slots} - 12)$ timeslots. This setting is intended to reflect the current situation of cloud computing, where some big companies provide their own services and many small consumers use services to execute their tasks.

Other parameters are set constant for the sake of simplicity. The quantity (throughput) of a service is 100 units for selling and one unit for buying. The valuation of a service is $\$1/(\text{slot} \cdot \text{unit})$ for selling and $\$3/(\text{slot} \cdot \text{unit})$ for buying. This setting means a loose supply-demand situation with no price competition, where the buyer's requirements are likely to be fulfilled. Table II shows the formulations. The simulation was conducted 10 times for each setting with different random seeds, and the average results are presented. The hardware and software configuration is identical.

2) Results

For the forward market, the desirable matchmaking time is less than the length of a timeslot because the allocation for the next timeslot must be determined within the current timeslot. For the spot market, it is preferable to finish the matchmaking as soon as possible, i.e. within one minute.

Figure 9 shows the overall runtime consumed by the market mechanism to perform a round of matchmaking. For the forward market, it takes more than five minutes with 720 timeslots and 1000 users. However, it is still shorter than the length of a timeslot, which we assume to be one hour or 12 hours. The result for the spot market is shown as "Number of Timeslots = 1". For the spot market, it takes less than one second. The overall runtime is essentially proportional to $|M| \times |N| \times |G| \times T$, which is the number of iterations to build the model and parse the results.

Figure 10 shows the runtime of the solver, i.e. excluding the time to build the model, etc. It takes less than 3 seconds in the worst case. The solver runtime is mainly affected by the difficulty to find the optimal solution, which is more sensitive to the number of conflicted orders than the number of timeslots.

The simulation results show that the proposed mechanism will scale beyond 720 timeslots, 1000 users, 10 providers and 10 services. In addition, the current implementation of the market mechanism is not intended to maximize the speed; it leaves room for improvement. Consequently, we conclude that the proposed mechanism will work practically with probable settings in the cloud computing environment.

VI. Conclusions and Future Work

We described a market-based resource allocation for cloud computing environments. It allows users to order an arbitrary combination of services from different providers. The forward market and the spot market run independently to make predictable and flexible allocations at the same time. The preliminary evaluation showed that the proposed mechanism worked with a realistic overhead under the probable settings of a cloud computing environment.

Our goal is to design an efficient public marketplace for cloud computing environments. We are interested in the autonomous behavior of the market price, particularly the interaction between the forward market and the spot market, where the forward price is expected to be a forecast of the spot price. An understanding of such behavior will help us to design and operate the cloud marketplace. We will investigate the market behavior by means of multi-agent simulations.

We are also interested in applying a market mechanism to enforce Green IT solutions. Our market model is aimed only at maximizing the total welfare in an economic sense, but this is not necessarily a desirable goal in an ecologic sense. Our future work will thus include optimization of energy consumption by means of a market mechanism.

References

[1] B. Schnizler, D. Neumann, D. Veit, and D. Weinhardt, "Trading grid services - a multi-attribute combinatorial approach," *European Journal of Operational Research*, vol. 187, no. 3, 2008, pp. 943-961.

[2] Z. Tan and J. R. Gurd, "Market-based grid resource allocation using a stable continuous double auction," *Proc. 8th IEEE/ACM Int. Conf. on Grid Computing (Grid 2007)* , 2007, pp. 283-290.

[3] L. Amar, J. Stosser, and E. Levy, "Harnessing migrations in a market-based grid OS," *Proc. 9th IEEE/ACM Int. Conf. on Grid Computing (Grid 2008)* , 2008, pp. 85-94.

[4] K. Hoki, "Outline of Japan Electric Power Exchange (JEPX)", *Transactions of the Institute of Electrical Engineers of Japan*, vol. 125, no. 10, 2005, pp. 922-925.

[5] H. Sato, Y. Koyama, K. Kurumatani, Y. Shiozawa, and H. Deguchi, "U-Mart: A Test Bed for Interdisciplinary Research in Agent Based Artificial Market", *Evolutionary Controversies in Economics*, 2001, pp. 179-190.

[6] M. A.Sattherthwaite, S. R.Williams, "The Bayesian Theory of the k-Double Auction", In: D. Friedman, J. Rust (Eds.), *The Double Auction Market - Institutions, Theories, and Evidence*, Addison-Wesley, Ch. 4, 1993, pp. 99-123.

[7] R. B. Myerson and M. A. Satterthwaite, "Efficient mechanisms for bilateral trading", *Journal of Economic Theory*, Vol. 29, No. 2, 1983, pp. 265-281.

[8] What is the 'itayose' method?, http://www.tse.or.jp/english/faq/list/stockprice/p_b.html , viewed Feb. 13, 2010.

[9] IBM ILOG CPLEX, http://www-01.ibm.com/software/integration/optimization/cplex/, viewed Feb. 13, 2010.

[10] lp_solve, http://tech.groups.yahoo.com/group/lp_solve/, viewed Feb. 13, 2010.