# Market-based Resource Allocation for Distributed Computing

Ikki Fujiwara[†], Kento Aida[†‡] and Isao Ono[*]

Market-based resource allocation is expected to be an effective mechanism to allocate resources in a cloud computing environment, where the resources are virtualized and delivered to users as services. In this paper we propose a market mechanism to efficiently allocate multiple computation/storage services among multiple participants, or the Cloud Service Exchange. The proposed mechanism enables users (1) to order a combination of arbitrary services in a co-allocation or a workflow manner, and (2) to receive future/current services at the forward/spot market. The evaluation shows that the proposed mechanism scales up to probable situations.

## 1. Introduction

Cloud computing is an emerging service paradigm for distributed computing environment. 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 to utilize cloud computing environment in their business. Advanced features related to QoS include performance guarantee of a service, co-allocation of multiple services and supporting a workflow organized by different services.

In the near future, we can expect that hundreds of providers compete to offer resource services and thousands of users also compete to receive the services to run their complex tasks on cloud computing environment with guaranteed QoS and limited budgets. However, an efficient recourse allocation mechanism among resource providers and users has not been well discussed.

In this paper, we propose a market-based resource allocation mechanism, which allows participants to trade their services effectively by means of the double-sided combinational auction. A market mechanism is one of the promising solutions to cope with the situation where a large number of participants, e.g. providers and users, trade the multiple kinds of resource services. The proposed mechanism enables the participants to trade future and current services in the forward market and the spot market, respectively.

## 2. Design Goal

This section briefly discusses the requirements for the proposed mechanism.

- **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. We employ the mixed integer programming method to strictly maximize the total welfare.
- **Predictability and flexibility**: Since the supply and demand in cloud computing environment change dynamically over time, the users may desire predictable allocation in advance as well as flexible adjustment in runtime. The proposed mechanism employs dual market mechanisms, the forward market for advance reservations of resources and the spot market for immediate reservations.
- **Combinational biddings**: The users may want to run complex tasks with advanced features, e.g. co-allocation. The proposed mechanism accepts combinational bids, with which the user can express complemental requirements for resource allocation.
- **Double-sided auctions**: In the proposed mechanism, both resource providers and users compete to offer/receive resources. Prices of resources are reflected by supply and demand of resources. The conventional commodities market mechanism [1] does not satisfy the requirement for the proposed mechanism. The proposed mechanism employs the double-sided auction model.

## 3. Related Work

Market-based resource allocation has been a hot topic in the grid literature for a decade. Schnizler et al. [1] introduced the double-sided combinational auction into grid service/resource allocation. In [1], resources are bundled by the resource providers and the users cannot select combination of resources. Tan et al. [2] proposed the Stable Continuous Double Auction (SCDA). It is not truly combinational, i.e. the users need to bid on multiple auctions in order to receive multiple resources. Amar et al. [3] illustrated a comprehensive grid market model including the futures market and the centralized/decentralized spot market. However, a detailed model of the futures market is not discussed.

While the computing resource market is not yet realized at the industrial level, the

[†] The Graduate University for Advanced Studies (SOKENDAI)
[‡] National Institute of Informatics
[*] Tokyo Institute of Technology

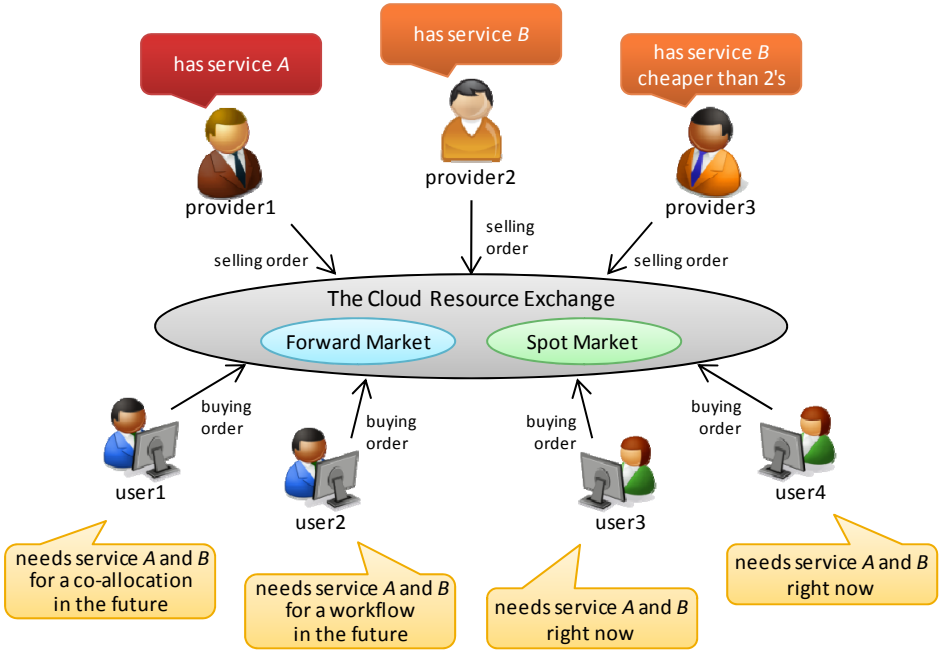


Fig. 1: Overview of the Cloud Resource Exchange

electricity market has been in practical operation for several years. For instance, Japan Electric Power Exchange (JPEX) started the operation in 2005. According to [4], it provides three markets: (1) the spot market for trading the electricity on the next day, (2) the forward market for trading the electricity delivered in some weeks or months, and (3) the forward bulletin board market for free transactions. Since the electricity and the computing service have similar nature (i.e. they cannot be stored), we regard the electricity market as a preceding model to the computing service market. However, the electricity market model cannot be directly applied to cloud computing because the electricity is almost uniform whereas the computing services vary in types and quality.

The stock market deals with a variety of stocks, which can be stored and reselled unlike the computing service. In this area, studies on dealing strategies and institutional design are carried out recently by means of multi-agent simulations. U-Mart [5] is a test bed for multi-agent simulations of the stock market, 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.

## 4. The Market Model

Figure 1 shows 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 resource provider/user agents participate to sell/buy the computing/storage resources abstracted as services. The participants interact with the spot market and the forward market independently using the bidding language described below.

Regarding the service we assume the following conditions:

- The amount of a service can be measured in a throughput (e.g. 60 requests/sec of service *A*). We use “unit” instead of “requests/sec” in the following part.
- From the provider’s point of view, a resource can be divided into arbitrary fraction (e.g. a resource of 60 units is divided into 20 units for *user1* and 40 units for *user2*).
- From the user’s point of view, a task can be divided into sub-tasks and executed on multiple resources (e.g. a task of 40 units is allocated a resource of 10 units from *provider1* and that of 30 units from *provider2*).
- Also, a task can be migrated during the runtime (e.g. a task running on a resource from *provider1* is suspended and resumed on that from *provider2*).

The proposed mechanism is characterized by three properties: (1) the bidding language defines the protocol between participants and markets, (2) the allocation scheme determines assignment of services, and (3) the pricing scheme fixes prices at which the participants trade their services. Below we formulate each property for each market.

### 4.1 Forward Market

The forward market deals with long-term advance reservations by means of the clearinghouse auction. It makes contracts periodically. A service is divided into timeslots, e.g. 1pm-2pm, and the timeslot is traded in the market. The market accepts orders from users any time but makes contracts every certain period, e.g. 3 hours.

#### 4.1.1 Bidding Language

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

A buying order from a user includes the following information:

- Valuation: The maximum price at which the user wish to buy the bundle of services.
- A bundle of arbitrarily services, each of which include:
  - Name: the kind of service
  - Quantity: the amount (throughput) of the service
  - Arrival: the earliest timeslot to start the task
  - Deadline: the latest timeslot to finish the task
  - Length: the total number of timeslots required to run the task

Note that valuation is given to a bundle, not to each particular service. In this way the user can express requirements for receiving multiple services, e.g. co-allocation or workflow. A contract is made if all services in the order are reserved for the user.

A selling order from a provider includes the following information:

- Valuation: the minimum price per timeslot at which the provider wish to sell the service
- Name: the kind of service
- Quantity: the amount (throughput) of the service
- Timeslot: the timeslot to provide the service

Note that a selling order includes only one service at one timeslot. The provider makes multiple orders for each service and each timeslot.

**Formulation**

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

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

where $q_{j,k}$ is the quantity of service $g_k$, $a_{j,k}$ is the arrival time, $d_{j,k}$ is the deadline and $l_{j,k}$ is the length. Similarly, a selling order is formulated as

$$S_i = (g_k, q_{i,k}, e_{i,k});\ 1 \le k \le |G|$$

where $e_{i,k}$ is the timeslot.

**4.1.2** Allocation Scheme

The allocation scheme determines a winner of an auction. We formulate the winner determination problem into a linear mixed integer program (MIP). Here we introduce four series of 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:

Maximize

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

s.t.

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

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

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

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

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

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

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

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

$$u_j \in \{0,1\}, \qquad 1 \le j \le |N| \quad (10)$$

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

$$z_{j,k,t} \in \{0,1\}, \qquad 1 \le j \le |N|, 1 \le k \le |G|, 1 \le t \le T \quad (13)$$

$$0 \le y_{i,j,k,t} \le 1, \qquad 1 \le i \le |M|, 1 \le j \le |N|, 1 \le k \le |G|, 1 \le t \le T \quad (14)$$

**4.1.3** Pricing Scheme

The pricing scheme calculates a price, which a provider/user actually earns/pays. The proposed mechanism employs the K-pricing scheme for pricing. This scheme intends to distribute the welfare generated by trading between the provider and the user. Let $0 \le K \le 1$ be an arbitrary fraction. The price $p$ is then calculated as $p_{i,j,k,t} = K(v_j z_{j,k,t}) + (1 - K)(v_i y_{i,j,k,t})$ for each resource and timeslot.

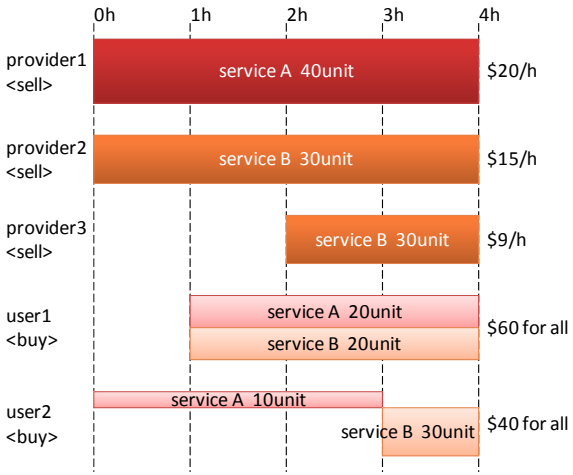


Fig. 2: Example orders in the forward market

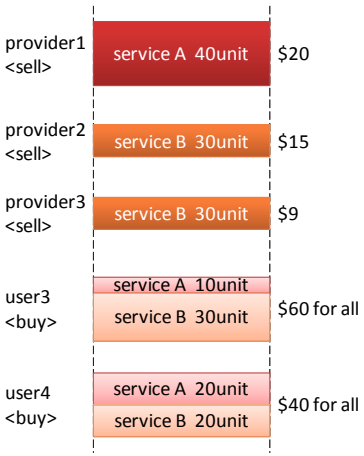


Fig. 3: Example orders in the spot market

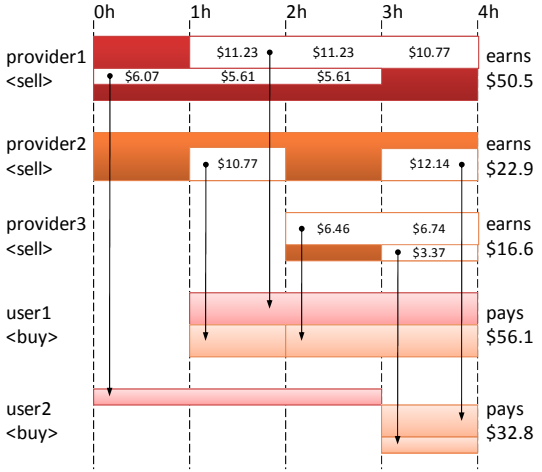


Fig. 4: Example allocation in the forward market

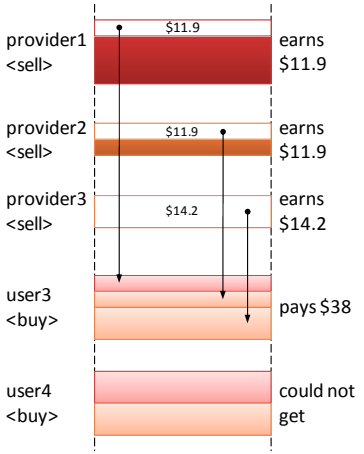


Fig. 5: Example allocation in the spot market

### 4.1 Spot Market

The spot market deals with short-term allocation by means of the continuous auction. It makes contracts continuously. The market matches orders whenever they come. The contracted service is allocated to the user within the current timeslot. The bidding language, the allocation scheme and the pricing scheme are almost same as those of the forward market except that they have only one timeslot.

### 4.2 Examples

Figure 2 and 3 illustrate examples of forward trading among five participants: *provider1* offers service *A*; both *provider2* and *provider3* offers service *B* with different prices; *user1* needs service *A* and *B* simultaneously; *user2* needs service *A* followed by *B*. As a result, all the users' needs are fulfilled. Note that *provider3* wins the competition for service *B* because the lower selling price makes more total welfare.

Figure 4 and 5 illustrate spot trading among participants. In this case *user4* loses the competition for service *B* because *user3* pays higher price. *Provider1* still has enough capacity for service *A*, but it is not allocated to *user4* since the order is combinational.

## 5. Simulator

We are developing a simulation environment, named W-Mart, to explore the market behavior by means of multi-agent simulations. Figure 6 shows the overall architecture of W-Mart. The Exchange and the agents are two main parts of W-Mart. The exchange has two market instances, namely the forward market and the spot market, independently. Each market mechanism is implemented on the top of MACE [1], which is a Java framework for combinational auction. The mechanism translates the orders into a mixed integer program

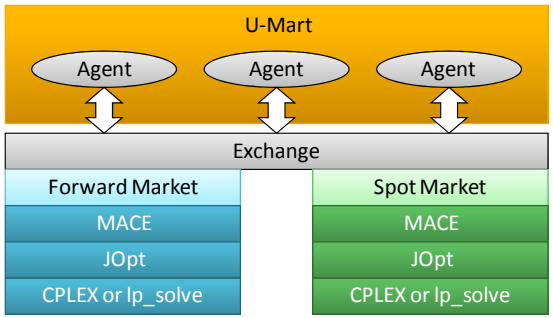


Fig. 6: Architecture of W-Mart

(MIP) using JOpt, which is a Java framework for MIP. JOpt abstracts the MIPs from the backend solver, which can be CPLEX or lp_solve.

The agents are to be implemented on the top of U-Mart, which is a Java framework for multi-agent simulations, especially focused on the futures trading of the stock market. U-Mart allows the machine agents and the human agents to trade simultaneously. We are planning to enable humans to participate in the service allocation, along with the algorithmic schedulers.

## 6. Evaluation

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

### 6.1 Experimental Setting

We carried out the stochastic 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 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 the spot trading and other cases represent the forward trading. The actual time span covered by timeslots depends on the length of the timeslot. For example, #slots = 720 represents 1 month with a timeslot of 1 hour, or represents 1 year with a timeslot of 12 hours. We refer to the example of the Japanese electricity exchange for the definition of the time granularity. We consider this extent of granularity is also applicable to the cloud computing environment.

The number of providers is fixed at 10, while the number of users has a range of {100, 400, 700, 1000}. Each provider offers each different service throughout the timeslots, i.e. all services are available anytime. Each user requires 1 to 5 services chosen randomly out of 10 services to be co-allocated. The task length varies from 1 to 12 timeslots. The task beginning time varies from 0 to (#slots − 12) timeslots after the ordering. 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 fixed for the sake of simplicity. The quantity (throughput) of a service is 100 units for selling and 1 unit for buying. The valuation of a service is \$1/(slot・unit) for selling and \$3/(slot・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 2 shows the hardware and software configuration to run the simulator. The simulation has been conducted 10 times for each setting with different random seeds and the average results are presented.

| | | |
|---|---|---|
| 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 |
| Beginning time of a Task | $0 \leq beg \leq T - 12$ , uniform distribution | $beg = 0$ 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 run | 10 times | |

Table 1: Simulation Parameters

| | |
|---|---|
| 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 |

Table 2: Simulation Environment

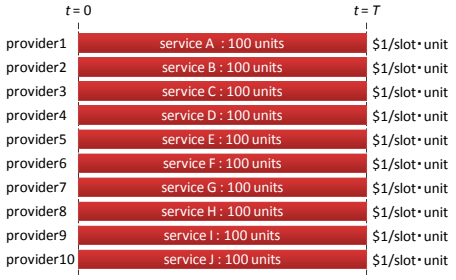


Fig. 7: Selling Orders in the Simulation

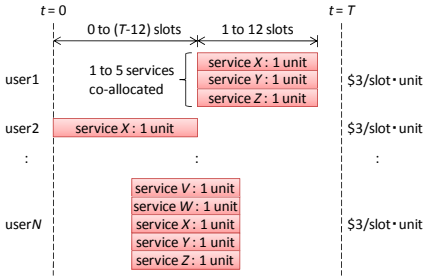


Fig. 8: Buying Orders in the Simulation

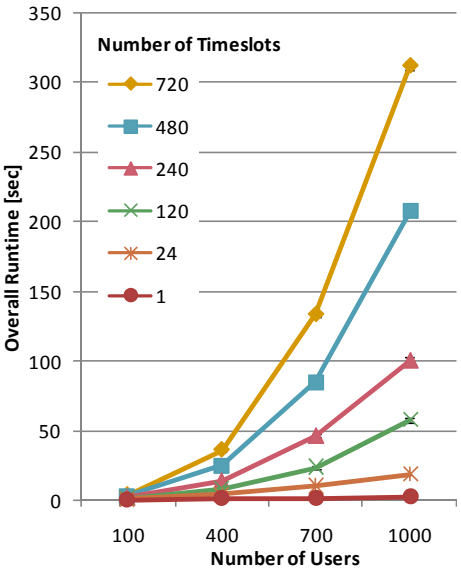


Fig. 9: Overall Runtime

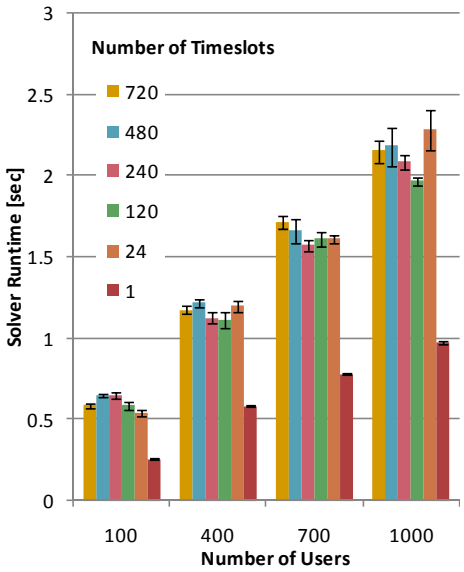


Fig. 10: Solver Runtime

### 6.2 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 1 minute. The result for the spot market is shown as “Number of Timeslots = 1”.

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 5 minutes with 720 timeslots and 1000 users. However, it will be still shorter than the length of a timeslot, which we assume to be 1 hour or 12 hours. For the spot market, it takes less than 1 second. The overall runtime is essentially proportional to $|M| \times |N| \times |G| \times T$, which is the number of iteration to build the model and parse the result.

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 number of variables in the model, which is rather proportional to the length of services and tasks 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 performance improvement. Consequently, we conclude that the proposed mechanism will work practically with probable situations in the cloud computing environment.

## 7. Conclusions and Future Work

In this paper we proposed the market-based resource allocation mechanism on cloud computing environment. It allows users to order an arbitrary combination of services to different providers. The proposed mechanism runs the forward market and the spot market independently to make predictable and flexible allocation at the same time. The evaluation showed that the proposed mechanism scales up to the probable situations in the cloud computing environment.

Our goal is to establish an efficient market-based resource allocation mechanism suitable for cloud computing. We are interested in the behavior of the exchange, particularly the interaction between the spot market and the forward market. We anticipate that a forward price shows a forecast of a spot price in the future. We are going to investigate the market behavior including such an interaction by means of multi-agent simulations.

## 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, pp. 943-961, 2008.

[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)*, pp. 283-290, 2007.

[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)*, pp. 85-94, 2008.

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

[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*, pp. 179-190, 2001.