Smart Cloud Marketplace - Agent-based Platform for Trading Cloud Services

Sergei Chichin, Mohan Baruwal Chhetri, Quoc Bao Vo, Ryszard Kowalczyk
School of Software and Electrical Engineering
Swinburne University of Technology,
Melbourne, Australia
{schichin, mchhetri, bvo, rkowalczyk}@swin.edu.au

Marcin Stepniak
PhD Studies, Systems Research Institute
Polish Academy of Sciences
Warsaw, Poland
stepniak@ibspan.waw.pl

***Abstract*—Cloud computing services are rapidly gaining popularity with more and more businesses actively migrating to the cloud, and many new cloud providers emerging. In such circumstances, there is a need for a market platform that allows for automated trading of cloud services between numerous independent users. Therefore, in this paper we propose Smart Cloud Marketplace (SCM) as a platform for trading cloud services based on intelligent agent technology. Software agents represent the cloud service consumers and providers in the marketplace, and make intelligent decisions on their behalf. The platform enables participants to use different trading policies in order to automate and improve the efficiency of resource trading. Moreover, SCM supports multimarket trading, where different market mechanisms can be deployed. We have used the SCM platform to test different market mechanisms developed within our research group. Our experimental evaluation demonstrates that Smart Cloud Marketplace is a useful, flexible and effective platform for conducting experiments and ultimately for trading cloud services.**

I. Introduction

In recent years there has been an exponential growth in the number of vendors offering cloud services with a corresponding increase in the number of enterprises looking to consume them. Gartner predicts that by 2016, cloud computing services will form the bulk of new IT spending [1]. The rapid growth in adoption of cloud services can be attributed to the numerous benefits it offers to businesses, such as reduced IT costs, scalability and collaboration efficiency [2]. Yet, there are several complex challenges associated with cloud service provisioning. The cloud environment is highly diverse and dynamic, with cloud providers offering diverse but flexible resource provisioning schemes, and consumers having highly variable workloads. This makes efficient resource allocation a very important and challenging problem. Recognizing this challenge, there has been a growing stream of interdisciplinary research including economics and computer science, with researchers proposing various market-based approaches for trading cloud services in different market settings, such as in [3][4][10][9]. However, at present, there is no marketplace platform that allows automated trading of cloud services between multiple independent users.

Hence, in this work, we design and develop Smart Cloud Marketplace - a prototypical platform for automated trading of cloud resources based on intelligent agents technology. The designed platform allows the market participants to choose from various deployed market mechanisms that are available in the system. The buyers and the cloud providers are represented by intelligent agents who make efficient trading decisions on their behalf. The market participants can employ different policies for automated resource procurement proposed in literature, such as in [6], as well as utilize their own policies to realize their custom trading strategies. The complex cloud marketplace environment where various individual users trade the resources in different markets makes intelligent decision making especially important. Moreover, intelligent agents are often used and prove themselves to be efficient in service offers search (e.g. setting up travel arrangements), or to bargain the price for specific product [7], which is similar to our problem.

Main contributions: We have proposed an agent-based platform called Smart Cloud Marketplace, which facilitates autonomous trading of cloud services between individual users. The platform supports multiple concurrent markets which operate various market mechanisms. It allows the buyers and sellers to employ their custom trading policies and to choose among different mechanisms. We have provided the reference architecture of Smart Cloud Marketplace and outlined two common interaction protocols in electronic markets. We have tested the Smart Cloud Marketplace platform in experimental setting. The considered experimental scenarios confirm feasibility and usefulness of the developed prototype that allows multiple users to trade cloud services in selected electronic markets. Our experiments also reveal that Smart Cloud Marketplace is a helpful tool for researchers and allows for testing and analyzing the different market mechanisms and trading policies under various scenarios. Moreover, Smart Cloud Marketplace facilitates construction of complex testing scenarios that involve multiple market mechanisms and trading strategies.

The rest of the paper is organized as follows. In Section II, we present the architecture of Smart Cloud Marketplace platform, and outline some modeled interaction protocols. In Section III, we discuss the market mechanisms and policy-based trading of cloud services. Section IV is dedicated to the discussion of considered experimental scenarios with the developed prototype. In Section V, we discuss relevant literature. We make concluding remarks in Section VI.

978-1-4799-4143-8/14 $31.00 © 2014 IEEE
DOI 10.1109/WI-IAT.2014.193

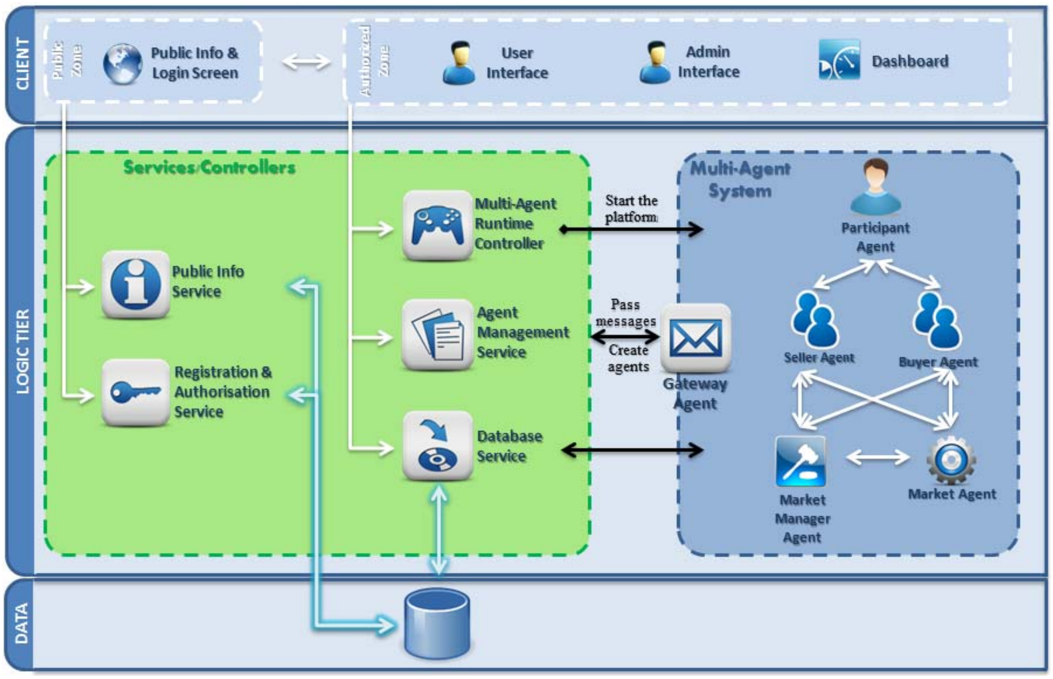


Fig. 1: Smart Cloud Marketplace Architecture

II. Smart Cloud Marketplace Platform

In this section we provide a high-level description of Smart Cloud Marketplace platform. We outline the platform requirements, describe the platform architecture, and describe the interaction protocols for common market types.

A. Background

A platform for trading cloud services must satisfy a number of important requirements:

- *Custom buyer/seller strategies:* Generally, cloud consumers and providers have varying and potentially conflicting aims and objectives and interact with one another in order to reach mutually acceptable agreements over the service usage terms and conditions. Hence, the cloud marketplace must allow them to use custom strategies to guide their interactions, when trading cloud services. The marketplace should provide the public information which allows for making informed trading decisions.
- *Multiple market mechanisms:* The cloud marketplace should support multiple market mechanisms for trading cloud resources. Similar to eBay's multiple auction types [8], the cloud marketplace should allow its users to concurrently trade resources in multiple markets using different market mechanisms.
- *Real-time trading:* As cloud computing delivers various IT services over the Internet, the cloud marketplace should allow users to buy/sell services remotely and in real-time.
- *Scalability:* Cloud computing is a very dynamic environment with a large number of users. Thus, the marketplace platform for trading cloud services must be scalable and should support a large number of buyers and sellers.

In order to address these requirements, we design a Smart Cloud Marketplace platform based on intelligent agents. Intelligent agents technology allows cloud buyers and providers to be represented by agents who make intelligent trading decisions on their behalf.

B. Platform Architecture

1) Overview: The reference architecture of Smart Cloud Marketplace is depicted at Figure 1. We briefly describe the main platform functional blocks below.

The market participants, i.e. buyers and sellers interact with the system via the web interface, which is divided into *Public Zone* with publicly available information, and *Authorized Zone* which allows users to participate in markets and trade cloud services. The system provides a *User Interface* for performing various market related operations, such as creating a market, joining a market, viewing market statistics, etc., and an *Admin Interface* for performing administrative operations such as market monitoring and management. The Smart Cloud Marketplace users can access the *Dashboard*, which provides the information about markets available for trade, as well as the general market statistics in order to be able to reason and choose the most interesting market, or to design a custom trading strategy.

The platform has a number of services that enable required user operations. *Multi-Agent System Runtime Controller* manages (e.g. starting, monitoring) the Multi-Agent System, which is the Smart Cloud Marketplace (SCM) Core. *Agent Management Service* provides communication channel (e.g. passes the messages, creates agents) between the services and the *Multi-Agent System*. *Database Service* is used to record all the market related information to the database and retrieve the required information and statistics for the *Dashboard* and intelligent agents.

2) Multi-Agent System (SCM Core): Our SCM Core architecture consists of six agent types that perform all the required functions of cloud market exchange. We employ broker-based trading model (i.e. use Market Agent as a mediator) in our system because such model achieves more fair matching for buyers and sellers [19]. We explain the role of each agent in the platform below.

Gateway Agent (GA) This agent is a main access point for all the requests coming from the user. GA serves as a communication gateway and is responsible for creating the agents, and passing the messages to these agents.

Participant Agent (PA) This agent represents a user of the system and it is responsible for managing and keeping track of all the Buyer and Seller Agents of this user. The main role of PA is to create Buyer and Seller Agents in order to achieve user-specific trade objectives. In the simplest form, PA would create a single Buyer or Seller Agent for trade. More sophisticated PA would create several agents in order to maximize the user's utility by coordinating the trade in multiple markets sequentially or simultaneously.

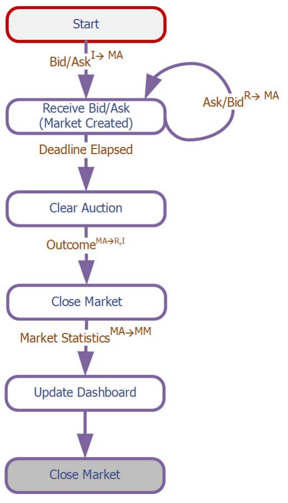
Buyer Agent (BA) This agent is in charge for submitting the buy requests i.e. bids to the required Market Agents. The BA can also request the Market Manager Agent to launch a Market Agent for the buyer's tender. The BAs can have a simple behavior (all the instructions come from the user) or it can operate based on more sophisticated policies (e.g. to select the most appropriate Market Agent, to choose the request prices, to apply different rebidding strategies).

Seller Agent (SA) This agent is similar to BA, except that it acts from the seller side, submitting sell offers to the Market Agents. It can also issue a request to the Market Manager Agent, to create a new Market Agent for its offer. The SAs can follow a simple behavior or can perform some sophisticated seller strategies (e.g. to make decisions about splitting the traded resource among multiple market mechanisms in order to maximize the generated profit).

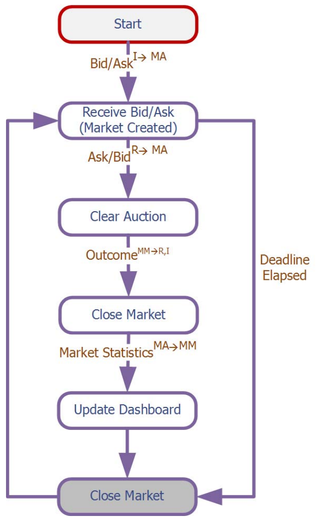
Market Agent (MA) This agent is responsible for finding the matches between the buyers' bids and sellers' offers based on a specific market mechanism. There may exist many different MAs that allocate and price the cloud services using various market mechanisms at the same time.

Market Manager Agent (MMA) This agent is responsible for managing all the MAs. Only a single MMA exists in the platform. This agent creates the MAs with selected market mechanisms upon the request of a BA or a SA.

3) Interaction Protocols: Typical markets include commodity markets, fixed-price (posted price) markets, bargaining (negotiation), tendering, and auctions [9]. Our platform is flexible enough to support multiple and different markets. Every market is characterized by its corresponding interaction protocol which defines the *rules of procedure* that every participant has to abide by. In this work, we present two interaction protocols for continuous and periodic auction markets (see Figure 2). It should be noted that these two protocols cover the most typical market mechanisms, including fixed-price, bargaining, tendering, and auctions. We model the interaction protocols as Finite State Machines (FSM), where the states represent the states that the market goes through and the transitions are triggered by message exchanges between the participants. In Figure 2, I refers to the Initiator, R refers to the Responder, *MA* refers to the Market Agent and *MM* refers to the Market Manager. Both *BA* and *SA* can play the roles of I and R.



(a) Periodic Auction Market



(b) Continuous Auction Market

Fig. 2: Interaction Protocol for Different Market Models

C. Implementation Details

The Smart Cloud Marketplace platform has been developed in Grails framework and has a web-interface implemented using Google Web Toolkit. The SCM *Multi-Agent System* platform core engine was developed in Java. It uses Java Agent DEvelopment Framework (JADE), which aims to simplify the implementation of multi-agent systems.

TABLE I: Notation

R	Set of VM type $\{1,\ldots,K\}$
s_r	Supplied amount of VMs type $r \in R$
o_r	Reserve price for the VM type $r \in R$
U	Set of buyers $\{1,\ldots,N\}$
a_{ur}	The amount of VMs of type $r \in R$ requested by buyer $u \in U$
d_u	The bundle of VMs requested by buyer $u \in U$
v_u	Declared valuation of the buyer $u \in U$
W	Social welfare
x_u	Allocation decision variable for buyer $u \in U$

TABLE II: Ask and Bid Model

Parameter		Services (k types) 1: *Small VM*	2: *Medium VM*	3: *Large VM*
Seller's Ask - $a = (\langle s_1,...,s_k\rangle, \langle o_1,...,o_k\rangle)$				
s_i	*Supply*	90	50	20
o_i	*Reserve price*	\$0.06	\$0.12	\$0.24
Buyer's Bid - $b_j = (\langle r_{1j},...,r_{kj}\rangle, v_j)$				
d_j	*Demand*	r_{1j} = 2	r_{2j} = 0	r_{3j} = 1
v_j	*Valuation*		\$1.0	

III. Smart Cloud Marketplace Examples

A. Market mechanisms for trading cloud services

Smart Cloud Marketplace allows to create multiple Market Agents, which operate different mechanisms. The researchers in academia and industry present various market-based approaches for cloud resource allocation and pricing, which could be added and operated in SCM. The cloud resource allocation with a unique resource provider has been addressed in [3][4][13][12]. The solution for double-sided cloud resource allocation is provided in [10][9][11]. Smart Cloud Marketplace facilitates the study, experimentation and analysis of new mechanisms. For example, our platform supported us in designing and testing the Greedy-RP [3] and Adaptive Greedy [4] market mechanisms for trading cloud services. In this section we consider a specific example of a single-sided market and describe the Greedy-RP mechanism, which was deployed in SCM for our conducted study.

1) Cloud Services Allocation Problem (CSAP): The problem notation is summarized in the Table I. An example of bid and ask in our mechanism design is depicted in the Table II.

A cloud provider owns a physical resource which is delivered to the buyers as a service in the form of Virtual Machines (VMs) of K different types. Similar to Amazon EC2 instance types, each VM type has its own pre-defined characteristics (e.g. amount of CPU, RAM) and price [15]. The seller declares the minimum possible trading prices, i.e. reserve prices, for each VM type, which are characterized by vector: $\langle o_1, ..., o_K\rangle$. The seller also specifies the amounts of supplied VMs of each type, which is defined as $\langle s_1, ..., s_K\rangle$.

We assume that N potential buyers participate in the auction $U = \{1,\ldots,N\}$. They have various cloud service requirements and the budget constraints, which are expressed in a form of a bid $b_u = (d_u, v_u)$, where $d_u = \langle a_{u1}, ..., a_{uK}\rangle$ is a bundle of required services and $v_u \geq 0$ is a bid valuation.

We formulate CSAP as an Integer Program as follows:

$$\text{Maximize } W = \sum_{u\in U} v_u x_u \tag{1}$$

Subject to:

$$\text{ARC: } \sum_{u\in U} a_{ur} x_u \leq s_r, \forall r \in R \tag{2}$$
$$\text{where } x_u = \{0,1\}, \forall u \in U$$

$$\text{RPC: } \hat{o}(d_u)x_u \leq v_u, \forall u \in U$$
$$\text{where } \hat{o}(d_u) = \sum_{r\in R} a_{ru} o_r \tag{3}$$

A standard objective in combinatorial auctions is to maximize the social welfare W, which is defined as a sum of winning buyers' valuations (1). The mechanism considers two constraints. Firstly, the Available Resource Constraint (ARC) implies that the amount of supplied resource cannot be exceeded (2), meaning that we cannot sell more than what is supplied. Secondly, the Reserve Price Constraint (RPC) ensures that the seller obtains as minimum the reserve price for the sold resources (3), meaning that the buyers can obtain the resource only if their valuation is at least as high as the bundle-specific reserve price $\hat{o}(d_u)$.

2) Greedy-RP Mechanism: The Greedy-RP mechanism was designed to address CSAP problem. It is based on greedy heuristic and consists of allocation and pricing schemes. The allocation scheme determines the users who will obtain the resources. The mechanism sorts all the specified bids in decreasing order based on their "price per item" value. Given the sorted order of the bids, the mechanism allocates the resources if both constraints (RPC and ARC) are satisfied. The pricing scheme aims to establish the prices that the users will have to pay for the granted resources. Greedy-RP is a discriminatory price auction, where each buyer pays his own price for the obtained bundle of resources, which is a common pricing model in combinatorial auctions. Greedy-RP determines critical-value prices, which are based on the "price per item" value of the first loosing competitor bid. Such price is the minimum possible price the buyer could have proposed in order to still be the winner of the auction. More detailed mechanism's description can be found in [3].

3) Public Information: Commonly, the electronic marketplaces reveal some limited information about the available markets. This information is essential for the buyers reasoning and strategic decision-making. It generally includes the static information about the market mechanism (e.g. market type) and market parameters (e.g. resource descriptions), and the market variables (e.g. market prices). In Smart Cloud Marketplace, the public information is published on the *Dashboard* and is available to the agents. What market information should be made public is a very challenging question. The market makers usually reveal the historical market prices. For example, Amazon EC2 Spot market provides the market uniform prices overtime. There is no uniform price in Greedy-RP (it is a discriminatory price auction), thus we determine the average P^r_{avg}, minimum P^r_{min} and maximum P^r_{max} prices for the VMs of different types, which we publish on the Dashboard. This information can be used by intelligent agents to achieve the required trading objectives by utilizing various agent trading policies, which are discussed in the next subsection.

B. Policy-based trading of cloud services

We use a policy-based approach [5][6] to automate market participation in Smart Cloud Marketplace. From an AI perspective, policies can be viewed as a form of guidance from humans that determine the actions of the software agents. They can be used to capture requirements and capabilities, and preferences over them in an expressive and flexible manner. They can also be used to capture strategic decision-making behavior to determine the actions of agents in different scenarios and contexts. The *PA* can use an internal reasoner to interpret the policies in its knowledge base and exhibit flexible and autonomous problem solving behavior with minimum human intervention.

In this section, we present some simple examples to illustrate how policies can be used to provide domain knowledge to the software agents participating in market. We consider the example of periodic auction market where the *SA* initiates the market by submitting an ask to the *MMA*. Based on the submitted ask, the *MMA* sets up a market which accepts bids from *BAs*. A potential *BA* has to make two key decisions - (a) should it participate in the market?, and (b) what bid should it propose? It can choose an appropriate action to execute based on the set of policies in its policy-base. To illustrate the policy-based decision making in Smart Cloud Marketplace we consider the following scenarios:

1) Scenario 1 - Minimize resource costs: Let us consider a scenario where the buyer wants to minimize the computing cost and job completion time is not a constraint. In this case the *BA* can select a conservative strategy to bid for resources. It can determine the bid price for each resource in the bundle based on the *minimum price per unit* based on previous history.

$$P_{max} = \kappa \cdot P^{r}_{min_n}, \text{ where } \kappa \leq 1 \tag{4}$$

where κ is a constant and $P^{r}_{min_n}$ is the *minimum price per unit* of the resource r for the last n rounds.

2) Scenario 2 - Minimize resource costs and have immediate access: In this scenario, the buyer wants to minimize resource costs and wants immediate access to the resources. The *BA* can choose a price which is closer to the *average per unit price* for each resource based on previous history and compute the price for the bundle from the per unit prices of each individual resource in the bundle.

$$P_{max} = \kappa \cdot P^{r}_{avg_n}, \text{ where } \kappa \leq 1 \tag{5}$$

where κ is a constant and $P^{r}_{avg_n}$ is the *average per-unit-price* of the resource r for the last n rounds.

3) Scenario 3 - Have immediate access to resources: In this scenario, in order to maximize the chances of getting access to resources immediately, the *BA* can determine a bid price which is closer to the *maximum price per unit* of each resource requested in the bundle.

$$P_{max} = \kappa \cdot P^{r}_{max_n}, \text{ where } \kappa \leq 1 \tag{6}$$

Based on the price per unit of each resource and the quantity requested, the *BA* can compute the price for the bundle.

TABLE III: Small Scale Experimental Setting

Cloud Provider Offer	Small Server	Medium Server	Large Server
Supply	4	4	4
Reserve Price	\$0.113	\$0.225	\$0.45

Buyer Requests	Small Server	Medium Server	Large Server	Valuation
Buyer 1	0	1	2	\$2.240
Buyer 2	0	3	0	\$1.195
Buyer 3	1	0	0	\$0.145
Buyer 4	0	0	1	\$0.737
Buyer 5	4	1	2	\$1.810

IV. Experimental Scenarios

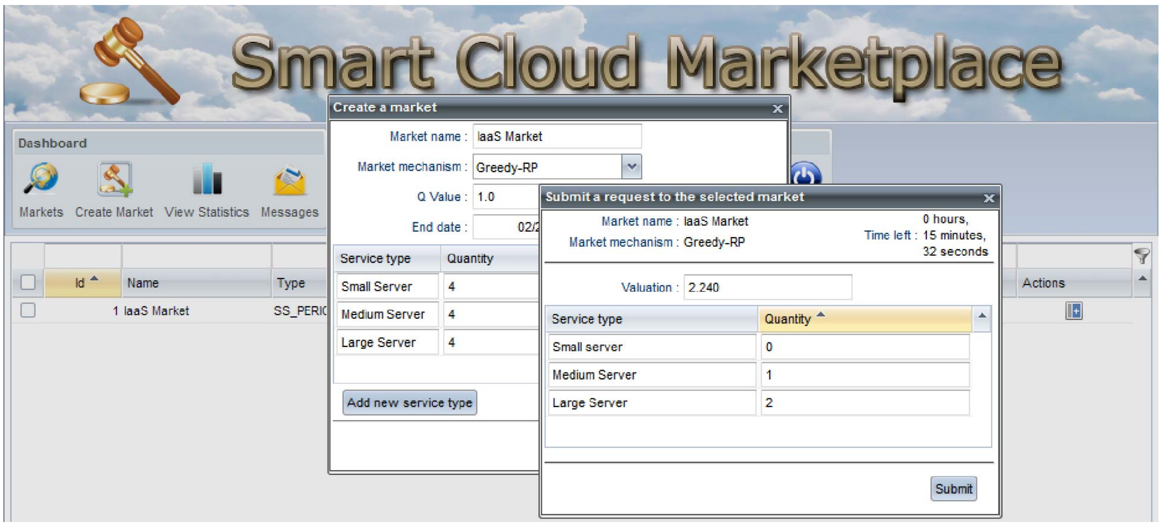
In this section we validate our developed platform under different experimental scenarios. We consider a small scale market case in order to demonstrate the workflow of the system and to study the impact of changing the individual behaviors, and a large scale resource trading scenario in order to study the dynamics of the market as a whole.

A. Small scale scenario

Our experimental setup (seller's offer and buyers' requests) is given in the Table III. We assume that there is a cloud provider, who wishes to trade the available cloud resource to the consumers, and selects Greedy-RP mechanism for services allocation and pricing. There are five customers interested in the provided resources. Each of them has the unique requirements about the needed services and corresponding prices that they are willing to pay.

The screenshot of the Smart Cloud Marketplace user interface is depicted at the Figure 3. We outline the standard workflow procedure for conducting the auction for resources. Initially, a cloud provider submits the request to create a new Market Agent for resource trading. The provider specifies all the required information about the supplied quantities, reserve prices, and the market specific characteristics, such as resource relative relation. When the corresponding market is created, it is available for cloud consumer's requests submissions at the Dashboard until the time expiry. The buyers follow a procedure to submit their requests to the market by filling in the form, where they specify the required resources together with their valuation for the requested bundle. Please note, that in this example, the buyers use the primitive agents for trading resources, but more sophisticated policies can be deployed. After the requests collection time expires, the market clears and returns the results to the participants.

The market users can also simulate different market behaviors and analyze the results in order to make better informed decisions when trading services in various market mechanisms. We imagine that the Buyer 1 decides to analyze the Greedy-RP mechanism and simulates different buyer behaviors. We assume that the Buyer 1 was honest when submitting the initial request (Table III). The buyer misstates the initial



Create a market

Market name : IaaS Market
Market mechanism : Greedy-RP
Q Value : 1.0
End date : 02/2

Service type	Quantity
Small Server	4
Medium Server	4
Large Server	4

Id	Name	Type	Actions
1	IaaS Market	SS_PERIC	

Submit a request to the selected market

Market name : IaaS Market
Market mechanism : Greedy-RP
Time left : 0 hours, 15 minutes, 32 seconds
Valuation : 2.240

Service type	Quantity
Small server	0
Medium Server	1
Large Server	2

Fig. 3: Smart Cloud Marketplace User Interface

TABLE IV: Mechanism Analysis: Misreporting Scenarios

N	Description	Requested Resource	Valuation	Win.	Price
0	Truthful Request	0, 1, 2	\$2.240	✓	\$1.125
1	Increased Valuation	0, 1, 2	\$2.500	✓	\$1.125
2	Less Resource	0, 0, 2	\$2.240	✓	\$0.900
3	Lower Valuation	0, 1, 2	\$2.000	✓	\$1.125
4	Much Lower Valuation	0, 1, 2	\$1.000	✗	-
5	More Resource	1, 1, 2	\$2.240	✓	\$1.238
6	Much More Resource	1, 2, 2	\$2.240	✗	-

true values and analyzes the impact. We present six different untruthful bidding scenarios in the Table IV.

The case 0 presents the buyer's honest request, which resulted in \$1.125 final price. The buyer wants to see if she is disadvantaged by the mechanism, if her valuation is higher or when less resource is requested, which we depict in cases 1 and 2, respectively. We can see that increased valuation does not affect the final price that the buyer pays. When less resource is requested, the corresponding final price is lower, which is also fair for the buyer. In the cases 3-6, the buyer wants to see if she can manipulate the mechanism to her benefit, by reducing the initial valuation or requesting more resource without changing the valuation. We can observe, that the buyer is not able to obtain a better price than in initial honest request, and by demanding more resources, the final price increases, which is a fair adjustment to the final price (case 5).

B. Large Scale Experimental Setting

Smart Cloud Marketplace allows to study the market dynamics by making changes to the market input. The developed prototype allows human users to bid on resources as well as enables creation of artificial demand and supply for simulation and analysis. To illustrate the usefulness of the platform we simulate different market scenarios. A cloud provider is trading its services for two days, conducting 10 auctions everyday. We assume that in the first day the market is balanced (demand is exactly equals to the supply), and the market demand during the second day is 20% higher than what is supplied by the cloud provider (resource is under-provisioned). The seller varies its reservation prices for the resources in order to come up with the best strategic behavior and analyses the market dynamics.

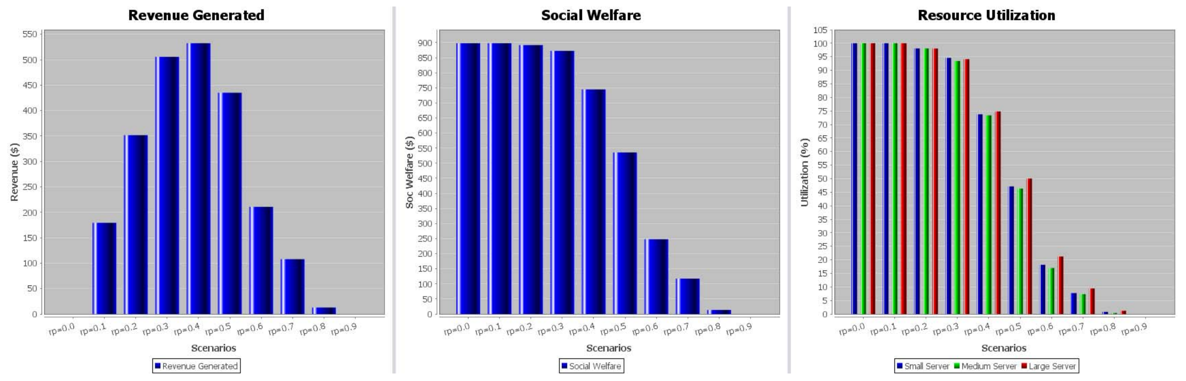
The seller records the following market statistics: the total generated revenue, total produced social welfare, and the individual resource utilization. The collected results are displayed in the Figure 4.

We know that the final prices that the buyers pay in GreedyRP are based on the competition for the requested resources. During the Day 1, the resource supply was equal to the demand, which resulted in low or little competition between the buyers. For that reason, the lower reservation prices ($rp < 0.5$) produced lower revenues for the seller (Figure 4a). During the Day 2, when there was not enough supply compared to demand, the seller's revenue generated with low reserve prices ($rp < 0.5$) was reasonable (Figure 4b). However, in both provisioning scenarios (exact and under-provisioning), if the seller is too greedy and the reservation prices are set too high, the revenue will reduce. The social welfare has a similar pattern in both cases: the higher the reservation price, the lower the social welfare. In terms of resource utilization, the lower the reservation prices the more resource is utilized. We can also notice that when the resource is under-provisioned, the resource utilization remains higher with higher values of reservation prices because of increased competition.

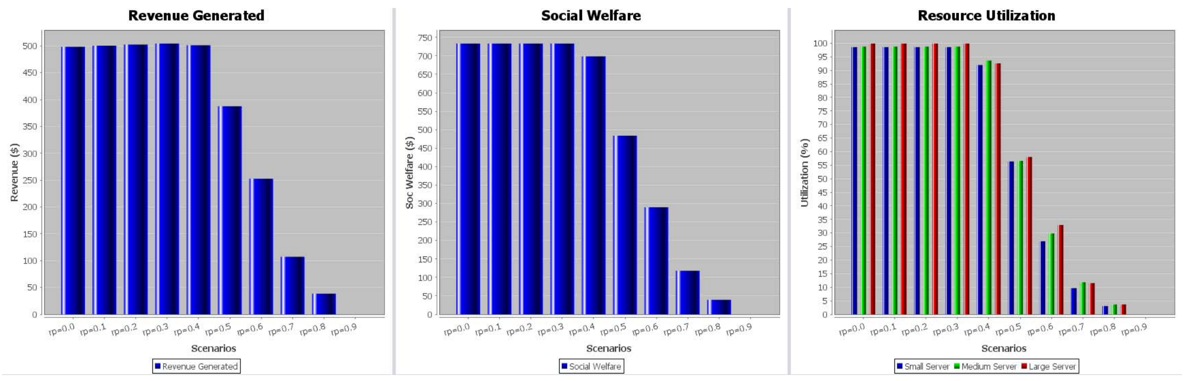
We have demonstrated that the Smart Cloud Marketplace platform is useful and allows for trading of cloud services, and for market analysis at the level of individual behavior change as well as to study the global dynamics of the market.

V. Related Work

Agent-based e-trading systems are frequently discussed in academia. The research papers mainly distinguish between two models, which are negotiation and brokering. In former model,



(a) Exact Resource Provisioning)



(b) Resource Under-provisioning

Fig. 4: Large Scale Scenario Results

TABLE V: Comparison between e-Markets presented in literature

	Agent-based System	Supported trading model		Multi-Market Support	Custom Trading Policy	Distributed Multi-user trading	Execution-time	
		Negotiation	Brokering				Real-time	Simulation
Smart Cloud Marketplace	✓	✓	✓	✓	✓	✓	✓	✓
Kasbah	✓	✓	✗	✗	✗	✓	✓	✗
METU-EMar	✓	✓	✗	✗	✗	✓	✓	✗
MAGMA	✓	✓	✓	✗	✗	✓	✓	✗
Kang and Han	✓	✓	✓	✗	✗	✗	✓	✓
Mandi	✗	✗	✓	✓	✗	✓	✓	✗
DRAGON-Lab	✗	✓	✓	✓	✗	✗	✓	✗
CloudSim	✗	✓	✓	✓	✗	✗	✗	✓

sellers and buyers negotiate prices directly with one another, in latter, there is a broker responsible for the selection of a suitable offer. Chavez et al. [16] propose Kasbah marketplace, where direct negotiation model with pro-active selling agents is applied. In their approach, the seller contacts the buyer agents in order to negotiate and find the best deal. The METU-EMar[17] marketplace facilitates resources trading on the Web. The resources are represented in a universal way. Initially, the market mechanisms match buying and selling agents who negotiate the final prices directly afterwards. Another marketplace which uses direct negotiations is MAGMA [18]. This system has a direct negotiation protocols as well as provides a Vickrey auction, which is a broker-based mechanism. Kang and Han [19] analyze both market models and conclude that direct transactions are not sufficiently effective and the broker who is responsible for matching the offers provides more fair results for both sellers and buyers.

To address the problem of market-based trading, the research papers also design the service-based (not agent-based) marketplaces. Garg et al. propose Mandi [20] which is a services-based framework for trading utility and cloud computing services. Mandi supports multiple negotiation protocols and allows its concurrent co-existence. Gao and Fan [21] propose two resource exchange models, namely market model, and tender/contract model, for an open Internet experimental platform for cloud DRAGON-Lab [22]. They experiment with the two models and discover that the market model had better performance than the tender/contract model in most of the testing tasks. The CloudSim [23] is a framework for modelling and simulation of cloud computing infrastructures and services. The authors claim that matching of service providers and customers is critical to cloud computing. Therefore, the framework has specific market-related properties associated to a simulation model to allow the cloud market modeling.

We summarize and compare the marketplaces described in literature in the Table V. Unlike the systems discussed above, our proposed platform supports broker-based markets as well as direct negotiations (can be mediated by broker, if required). The main advantage of transactions with a broker is the certain, pre-established, fair rules. The second differentiation point of our marketplace is that it allows the individual users to be represented by the intelligent agents and utilize their custom trading policy to achieve their individual trading objectives, which was not addressed in any of the works above. Finally, SCM is a system aimed for real-time trading, which also supports experimentation in a simulation environment to allow researchers analyze their proposed mechanisms and trading policies.

VI. Conclusion and Future Work

The fast growing usage of cloud computing has crated an increasing need for the cloud marketplace that would allow the cloud providers and consumers trade services as independent users. Therefore, we have proposed Smart Cloud Marketplace, which is an agent-based platform for automated trading of cloud services. The designed prototypical platform (1) supports multiple concurrent markets which operate based on different mechanisms, (2) allows the market participants to be represented by intelligent agents who make efficient trading decisions on their behalf achieving users' individual trading objectives, (3) and automates trading of cloud services between the distributed users in real and simulated time. We have conducted several experimental scenarios that confirmed that the platform allows geographically distributed users to trade cloud services in real-time. Our experiments have also revealed that the Smart Cloud Marketplace platform is useful for designing new market mechanisms and trading policies. It helped our research group to experimentally analyze the designed market mechanisms for trading cloud resources. In particular we have been able to demonstrate truthfulness of the designed mechanism and analyze the impact of the varying reservation prices on such market mechanism metrics as generated revenue, social welfare, and resource utilization.

In future, we plan to add more market mechanisms and buying/selling policies to the platform, and conduct various experiments in order to analyze and compare their effectiveness and efficiency. Another important investigation will focus on the mechanisms for Participant Agent in order to enable coordination of multiple Buyer and Seller Agents with the objective to maximize the participant's utility from trade.

Acknowledgment

This work was partially funded by the Smart Services CRC, which is proudly supported by the Australian Federal Government's CRC Grant Program. Marcin Stepniak is partially supported by the Foundation for Polish Science under International PhD Projects in Intelligent Computing. Project financed from The European Union within the Innovative Economy Operational Programme 2007-2013 and European Regional Development Fund.

References

[1] Gartner, Inc., "Gartner Says Cloud Computing Will Become the Bulk of New IT Spend by 2016", Feb. 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2613015

[2] T. Velte, A. Velte, R. Elsenpeter, *Cloud Computing, A Practical Approach*. McGraw-Hill Osborne Media, 2009, ch. 2.

[3] S. Chichin, B. Q. Vo, and R. Kowalczyk, *"Truthful Market-based Trading of Cloud Resources with Reservation Price"*, in IEEE *SCC*, 2014.

[4] S. Chichin, B. Q. Vo, and R. Kowalczyk, *"Adaptive Market Mechanism for Efficient Cloud Services Trading"*, in IEEE *CLOUD*, 2014.

[5] M. B. Chhetri, Q. B. Vo, R. Kowalczyk, *AutoSLAMA policy-based framework for automated SLA establishment in cloud environments*, in Concurrency and Computation: Practice and Experience, doi: 10.1002/cpe.3171 (2013)

[6] M. B. Chhetri, Q. B. Vo, R. Kowalczyk, *Policy-based automation of SLA Establishment for Cloud Computing Services*, in IEEE/ACM CC-Grid, Ottawa (Canada), 13-16 May 2012

[7] N. R. Jennings, M. Wooldridge, *Agent Technology*, Springer Berlin Heidelberg, 1998, ch. 1.

[8] "eBay Seller Center", March 2014. [Online]. Available: http://sellercentre.ebay.com.au/

[9] I. Fujiwara, "Study on combinatorial auction mechanism for resource allocation in cloud computing environment," Ph.D. dissertation, The Graduate University for Advanced Studies: (SOKENDAI), 2012.

[10] J. Roovers, "A market design for iaas cloud resources," Master's thesis, University of Antwerp, 2011.

[11] J. Stößer, D. Neumann, and C. Weinhardt, "Market-based pricing in grids: On strategic manipulation and computational cost," *European Journal of Operational Research*, vol. 203, pp. 464–475, 2010.

[12] M. M. Nejad, L. Mashayekhy, and D. Grosu, "A family of truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds," in *IEEE Cloud*, 2013, pp. 188–195.

[13] S. Zaman and D. Grosu, "Combinatorial auction-based allocation of virtual machine instances in clouds," in *CloudCom*, 2010, pp. 127–134.

[14] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004, ch. 9.

[15] "Amazon ec2 instance types," Jan. 2014. [Online]. Available: http://aws.amazon.com/ec2/instance-types/

[16] A. Chavez and P. Maes, "Kasbah: An agent marketplace for buying and selling goods," in *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, 1996, pp. 75–90.

[17] A. Dogac, I. Durusoy, S. Arpinar, E. Gokkoca, N. Tatbul, and P. Koksal, "Metu-emar: An agent-based electronic marketplace on the web," in *Research and Advanced Technology for Digital Libraries*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, vol. 1513, pp. 777–790.

[18] M. Tsvetovatyy, M. Gini, B. Mobasher, Z. W. Ski, and W. Ski, "Magma an agent based virtual market for electronic commerce," *Applied Artificial Intelligence*, vol. 11, no. 6, pp. 501–523, 1997.

[19] N. Kang and S. Han, "Agent-based e-marketplace system for more fair and efficient transaction," *Decision Support Systems*, vol. 34, no. 2, pp. 57–165, 2003, agents and E-Commerce Business Models.

[20] S. Garg, C. Vecchiola, and R. Buyya, "Mandi: a market exchange for trading utility and cloud computing services," *The Journal of Supercomputing*, vol. 64, no. 3, pp. 1153–1174, 2013.

[21] F. Gao and T. Fan, "Could resource exchange model for dragon-lab," in *The 2nd International Conference on Cloud-Computing and Super-Computing*, ser. ASTL, vol. 22. SERSC, 2013, pp. 196–200.

[22] J. Wang, Z. Li, G. L, C. Jiang, X. Li, and Q. Zhang, "Dragon-lab–next generation internet technology experiment platform," *Science in China Series F: Information Sciences*, vol. 51, no. 11, pp. 1908–1918, 2008.

[23] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.