2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

Continuous Double Auction in Grid Computing: An Agent Based Approach to Maximize Profit for Providers

Aminul Haque[1], Saadat M. Alhashmi[2]
School of Information Technology
Monash University Sunway Campus
Bandar Sunway, Malaysia
[1]aminul.haque@infotech.monash.edu.my
[2]saadat.m.alhashmi@infotech.monash.edu.my

Rajendran Parthiban
School of Engineering
Monash University Sunway Campus
Bandar Sunway, Malaysia
rajendran.parthiban@eng.monash.edu.my

Abstract— **Economic models are found efficient in managing heterogeneous computer resources such as storage, CPU and memory for grid computing. Commodity market, double auction and contract-net-protocol economic models have been widely discussed in the literature. These models are suitable for sharing distributed computer resources that belong to different owners. Agent technology can be used to manage these heterogeneous resources without human intervention, since agents are autonomous and intelligent in behavior. In this paper, we develop and simulate an agent-oriented double auction economic model. We compare the performance of our agent-oriented model with traditional double auction model, and show that the agent-oriented model is good in maximizing profit for providers.**

I. INTRODUCTION

Grid computing is a promising platform to solve computationally intensive problems in science, engineering and technology. Grid computing is termed as a promising platform due to three reasons [1] – (a) it is more cost-effective than traditional computing, (b) it provides a significant amount of computing power to solve complex problems and (c) dispersed resources can be shared cooperatively toward a common objective.

Economic approaches are suitable for managing grid resources [2, 3]. Price is an important factor that allows users belonging to different organizations to access grid resources legally. Along with better management of grid resources, economic approaches can provide incentives such as profit and ability to negotiate. Double auction is one of the widely proposed models for grid computing. It is well known in this context due to its ability in handling large number of users, maintaining market equilibrium and minimizing communication overhead [4, 5].

Agents are defined as "*a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives* [6]". Due to the autonomous characteristic of agent technology, it could benefit the grid in dynamically negotiating and managing large-scale heterogeneous resources. In the grid computing, this negotiation could be based on several attributes such as price, quality of service and deadline.

This paper focuses on how to maximize profit for providers by integrating agent characteristics (such as decision making) with double auction economic model. In this model, user-agents and provider-agents work autonomously on behalf of the users and providers, respectively. We use continuous double auction where the auction process continues until there is at least one user or one provider in the simulated market. In our auction mechanism, we develop a provider-agent which is able to sense runtime rejection rate (rate of failure auction) of jobs and update the pricing accordingly to prevent from maximizing rejection rate.

The rest of the paper is organized as follows. Section 2 presents some related work in double auction model in grid computing. Section 3 explains our proposed framework. Section 4 describes the implementation of our proposed framework with agent technology. Section 5 discusses the simulation setup and ends with an evaluation and discussion.

II. LITERATURE SURVEY

In Double Auction model (DA), users post their requests and service providers their offers. If a user's request matches with a provider's offer, the trade is executed. There are two types of DA in the literature. The first type is the periodic DA in which the auction has a predefined time frame during which participants can submit their bids and asks. The second type is Continuous DA (CDA) in which users and providers can submit their bids and asks at any time during the trading phase. In this paper, we focus on CDA, since it is more flexible for submitting requests and offers.

Marek et al. [7] apply CDA for workflow scheduling in grid resource allocation. They propose different workflow execution strategies such as resource selection, resource valuation for users and providers, and bidding strategy to help users to do a fast and cheap execution of the workflow. They assume that their users have unlimited budget so that, they can focus on minimizing the execution cost. However, in real grid scenarios, the proposed model may not be applicable for users with budget constraints. Their strategy to update provider-bids is based on auction success ratio. In our model, we update based on supply and demand, which is more appropriate in market-oriented grid computing.

Tan et al. compare SCDA (Stable Continuous Double Auction) with CDA (Continuous Double Auction) [2]. A Compulsory Bidding Adjustment Layer (CBAL) to identify an appropriate bid for a current market is added to CDA. Hence, it is called SCDA. They have shown SCDA as superior to the CDA in terms of economic efficiency and

978-0-7695-4191-4/10 $26.00 © 2010 IEEE
DOI 10.1109/WI-IAT.2010.105

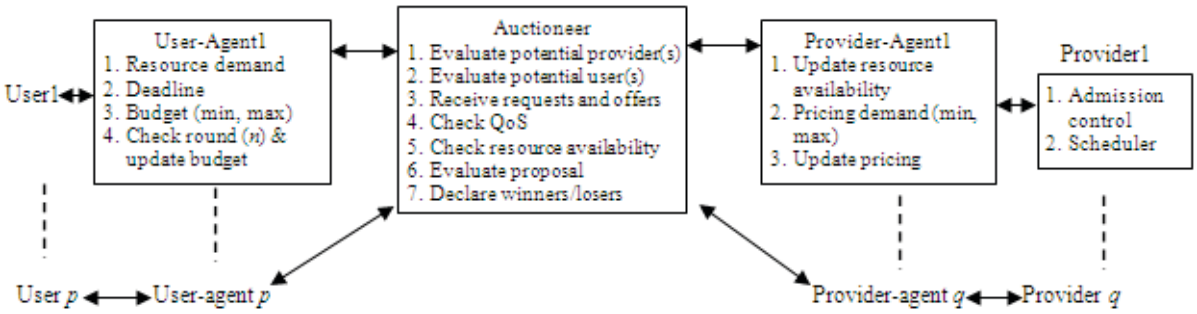

Figure 1: Proposed framework of agent-oriented continuous double auction

scheduling efficiency. However, they have conducted their experiments with a few requests and offers. Therefore, we do not really know whether their conclusions are valid for a large scale grid computing.

Pourebrahimi et al. [8] propose CDA against traditional non-market based resource allocation and find market-based allocation more efficient than simple FCFS (First Come First Serve) mechanism in terms of resource and task utilization. Their proposed pricing function is able to reflect market price based on supply and demand. The focus of our work is to use supply and demand and to develop a provider agent to maximize profit for providers.

III. PROPOSED FRAMEWORK

Our framework is built on three main parts; user-agent (UA), provider-agent (PA) and auctioneer (Figure 1). These parts are presented below with a brief explanation.

A. User-Agent

A UA is supposed to be set by a user itself. Typically, a user sets his/her UA with resources such as storage and number of CPUs to execute his/her job. In addition, a user sets a deadline to get the job done and a minimum and maximum budget he/she is willingly to pay for the job. The auction happens in multiple rounds and the current round number is denoted by n. The last action by the UA is to check the round of auction and update budget, if necessary.

B. Provider-Agent

We develop an algorithm so that a PA can update its pricing based on supply and demand during runtime. To define the algorithm, we initiate a parameter called f which tells how frequently a provider would like to update his resource prices.

$$cpr = \frac{initial\ resource\ amount - current\ resource\ amount}{prc}$$

Where,

$$prc = \frac{initial\ resource\ amount}{100}$$

The parameter cpr refers to current percentage of a particular resource. Once a provider-agent gets its cpr, it can check whether the resource prices have to be updated or not using the following equation,

$$cnf = \frac{cpr}{f} = \frac{current\ rate\ of\ change\ in\ resource}{frequency\ to\ change\ resource\ price}$$

if [$cnf >= 1$]
[*Change the resource price*]
else [*Do not change the resource price*]

C. Auctioneer

Auctioneer performs several crucial tasks to precede the auction. At first it evaluates and lists potential users and providers, who would like to join the auction. A user and a provider can be removed from their respective list, if the user's maximum budget is reached and still fails in all previous rounds, and the provider does not have enough resources to serve at least one user. Auctioneer then starts measuring a proposed request (starts from the maximum bidder) and checks whether it matches with an available offer (stars from the minimum asker) or not. In our work, we develop the PA using some strategies, so that the agent can make decision to maximize profit for providers.

Provider-Agent
1. Update resource availability
2. Pricing demand (min, max)
3. Measure job-rejection-rate
4. Decide method to update pricing
5. Update pricing

Figure 2. Amended provider-agent

Figure 2 shows the structure of our proposed PA. Here, two actions have been added. One is to measure the job-rejection rate by the PA dynamically during runtime. Job-rejection-rate is explained in the next section. Then, decide a method to update the pricing based on the measured rejection-rate. Here, we apply two methods to update pricing; one is based on supply and demand, as aforementioned, and the other one does not consider supply and demand but relaxes pricing based on user-budgets. The following algorithm shows how prices are relaxed:

Algorithm to choose a pricing method:
if (*measured job-rejection-rate > job-rejection-rate set by a provider*)
[*relax pricing based on user-budget*]
else [*follow supply and demand-based pricing*]

To let the algorithm work, a provider is supposed to predefine a rejection-rate. A question arises: "*How to set the best rejection-rate to get more revenue from the auction?*". The focus of this paper is to find the answer for this question through a comprehensive simulation with different number of users and providers.

IV. IMPLEMENTATION

We implement the proposed model using a cross-platform multi-agent programmable modeling environment known as Netlogo [9], [10]. We choose Netlogo because:

- Netlogo is a FIFA (Foundation for Intelligent Physical Agent) conformant platform [11]
- It has extensive built-in models to deal with multi-agents
- It can work as a 'simulated parallel' environment [9]
- It is platform (Mac, Windows, and Linux) independent [10]

A. Agent Interaction and Output

In the Netlogo framework, three different results can be obtained based on the interaction of UA and PA with the auctioneer. The first result describes the job rejection rate for a provider. Job rejection occurs due to scenarios such as disagreement of resource prices or unavailability of resources. This rate is calculated using two parameters - the total number of rejected jobs ($n_{rejected}$) and the total number of requested jobs ($n_{requested}$). The job rejection rate is assumed to range from 0 to 1. The job rejection rate, R_{rate}, is:

$$R_{rate} = \frac{n_{rejected}}{n_{requested}} \qquad (1)$$

The second output demonstrates the total revenue earned by a provider. It sums only the prices of the accepted jobs. Hence, the total revenue, T_{rev}, is:

$$T_{rev} = \sum_{i=1}^{j} M_i \qquad (2)$$

Where i denotes the executed job number, j denotes total number of executed jobs and M_i defines agreed price (between a user and a provider) for the i^{th} executed job. The third output presents how the resources on provider side are being utilized. In this paper, we consider the utilization of storage and CPU. The percentage of utilization, U_{res}^{m} of any resources can easily be calculated using the following equation:

$$U_{res}^{m} = \frac{initial\ resource_{res}^{m} - available\ resource_{res}^{m}}{initial\ resource_{res}^{m}} * 100 \qquad (3)$$

Where, m refers to the utilization (U) of a specific resource of type res.

V. SIMULATIONS and RESULTS

We use the following parameters in Table 1to simulate the performances of the traditional and agent-based CDA models.

TABLE 1. RESOURCE CONFIGURATION

User/provider-level parameter	User-level-range	Provider-level-range
Storage/diskspace (GB)	500-1000	60000-60500
Number of CPUs (MIPS per CPU)	5-10	800-850
Minimum Budget/demand ($)	100-200	100-200
Maximum Budget/demand ($)	200-400	200-400

Column 1 of Table 1 represents different parameters that a user and a provider use to set their agents. Since the total number of users and providers in our model is large, we select a value from the ranges shown in columns 2 and 3 automatically. In addition, to simplify simulation, we assume concurrent arrival of different requests and offers.

In order to find the best rejection-rate for more revenue, we conducted a comprehensive simulation for the following cases;

Case (I) 1000 users and 50 providers
Case (II) 1000 users and 10 providers
Case (III) 100 users and 50 providers
Case (IV) 100 users and 10 providers

We use the algorithm in Section 3.3.2 for a few providers. The rest of the providers use traditional CDA. We vary the rejection rate from 0 to 1 for each case to find the rate, which gives the maximum profit. We performed many simulations. In this paper, we only presented the average results for job-rejection rate, revenue and utilization of resources (storage and CPU). Using the results, we compare two types of providers;

(i) one type that uses traditional CDA and denoted by Traditional Provider (TP) and
(ii) the other type that uses our agent-based algorithm and denoted by Intelligent Provider (IP).

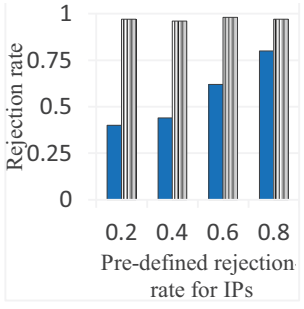

Figure 3. Job rejection rate

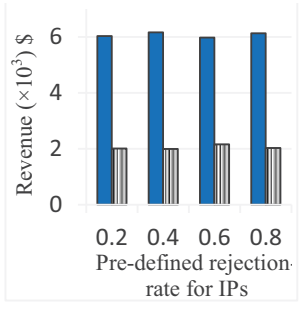

Figure 4. Revenue

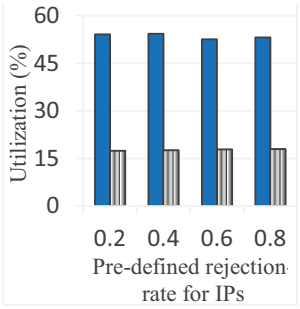

Figure 5. Utilization of diskspace

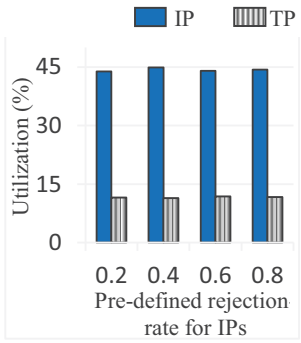

Figure 6. Utilization of processor

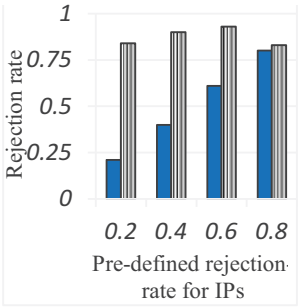

Figure 7. Job rejection rate

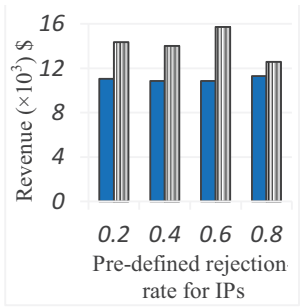

Figure 8. Revenue

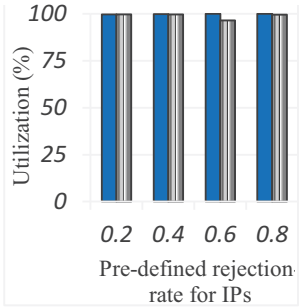

Figure 9. Utilization of diskspace

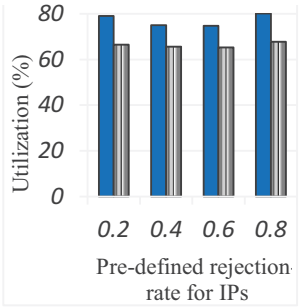

Figure 10. Utilization of processor

Case (I) 1000 users and 50 providers

The results obtained for Case I is shown in Figures 3-6. In this case, ten are IPs and the rest are TPs. Hence, the predefined rejection-rates along X-axes are only applicable for IPs. In terms of job rejection rate (Figure 3), IPs perform better than TPs. The overall rejection rate of TPs is always high, since they have no mechanism to control the rejection rate. IPs have the mechanism to control the rejection rate. The control mechanism tells us when and how to relax pricing. For example, if a predefined rejection rate is 0.4, the corresponding PA only relaxes pricing when the rejection rate exceeds 0.4 during runtime. For some pre-defined rejection rates (e.g. 0.2, 0.4), the average rejection rates obtained are higher than the pre-defined values. Most of the IPs do not get any job or get few jobs, since other IPs have accepted all the jobs. For this reason, the average rejeciton rate for these values is higher than the predefined rejection rate. The actual rejection rates are closer to pre-defined rejection rates for increasing pre-defined rejection rates. If the pre-defined rejection rate is high, the chance to get jobs increases for most IPs. This drives the average rejection rate closer to the pre-defined rejection-rate.

Figure 4 compares the revenue of IPs and TPs. The revenue for IPs is greater than that of TPs. In terms of IPs, the overall revenue is not affected by the predefined rejection rates. When the predefined rejection rate increases, it increases the chance of getting some revenue by all the IPs, but it does not increase the chance of getting more users in total compared to that of lower predefined rates. For instance, if we consider a predefined rejection rate of 0.2, only a few IPs would be enough to accommodate more jobs and the rest of the IPs do not get any jobs or only get a few jobs. If the predefined rejection rate is 0.8, a few IPs take time to relax their pricing, which helps rest of the IPs to get jobs. For these same reasons, there is not much variation in revenue for TPs.

Figure 5 and Figure 6 present the utilization of diskspace and processor respectively. The utilization of resources for IPs is better than that of TPs. The reasons given for Figure 4 are sufficient to explain the results in Figures 5 and 6.

Based on the above discussion, we recommend a rejection rate of 0.8 for Case I. Although this rejection rate does not ensure increased profit, it at least ensures some jobs for all IPs.

Case (II) 1000 users and 10 providers

In this case, IPs have less job rejection rate than TPs as shown in Figure 7. Unlike in Case I, in this case, the demand (number of users) is greater than the supply (number of providers). Hence, even for low predefined rejection rates (such as 0.2 and, 0.4), the IPs that get the jobs earlier cannot accommodate all the jobs. Therefore, the rest of the IPs get chance to accommodate the left over jobs and helps to make the average rejeciton rate less. Due to excess demand compared to supply, TPs earn more revenue than IPs at all predefined rates (Figure 8). In our algorithm, IPs relax prices based on predefined rates. This pushes IPs to accept jobs with relaxed prices and quickly utilize all the resources. On the other hand, due to excess demand, the TPs continue to receive and accept jobs based on supply and demand. This helps them to get more revenue. The same explanation can be imposed for resource utilization as well (Figure 9 and 10).

We recommend no predefined rejection rates for this case, since demand is higher than the supply. Hence, it would be good for IPs to follow the method of TPs for getting competitive revenue.

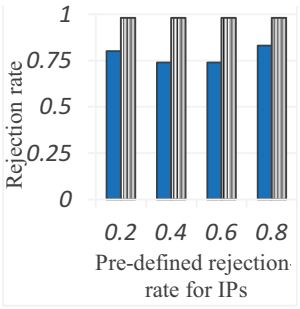

Figure 11. Job rejection rate

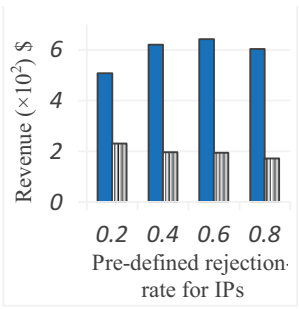

Figure 12. Revenue

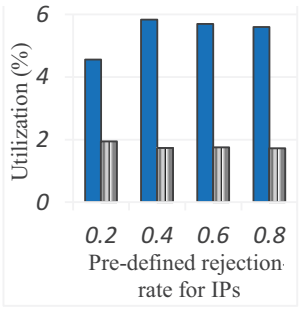

Figure 13. Utilization of diskspace

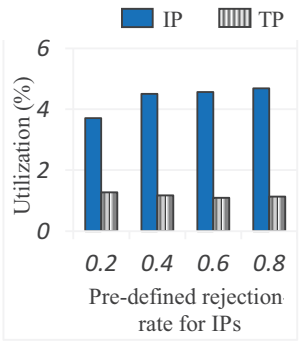

Figure 14. Utilization of processor

Case (III) 100 users and 50 providers

In this case, ten providers are IPs and the rest are TPs. We present the average results of 10 IPs and 40 TPs below for brevity. Figure 11 presents the comparison of job rejection rates for IPs and TPs. Since the supply (number of providers) is greater than demand (number of users), most of the providers remain unutilized. This causes high rejection rate for both IPs and TPs. The overall rejection rate is roughly constant for TPs. For IPs, this first decreases and then increases. We define two conditions that help to discuss this interesting trend in rejection rate. The first property is *"acceptation of jobs by more IPs"* and the second one is *"quick acceptation of jobs"*. If both of them are satisfied, job rejection rate is less. If the predefined rejection rate is 0.2, the second condition occurs, since prices are quickly relaxed to get jobs. When the predefined rate is 0.8, the first condition occurs. For this rate, it takes time to relax prices. This allows the other IPs to get jobs. For these reasons, the average rejection rates at 0.2 and 0.8 are high. For the predefined rates 0.4 and 0.6, both conditions occur and help to reduce the rejection rate.

The trend in Figure 12, which shows the revenue, can be explained based on the trend in Figure 11. Revenue of IPs is higher than that of TPs. Revenues at predefined rates 0.4 and 0.6 are higher than those at 0.2 and 0.8. Figure 13 and Figure 14 show the comparison of resource utilizations. The overall resource utilization is very less, since there are only 100 jobs.

For the values in the range [0.4, 0.6], the rejection rate is less, because both the conditions are satisfied, which were used to explain Figure 11. Due to less rejection rate, more revenue can be achieved. Hence, we recommend a predefined rejection rate in the range of [0.4, 0.6] for this case.

Case (IV) 100 users and 10 providers

For this case, we get similar results as in Case III. Based on the results and discussions presented thus far, we can answer the question raised in Section III using Table 2.

TABLE 2. SCENARIO-BASED PREDEFINED REJECTION RATE FOR INTELLIGENT PROVIDERS

Case	Recommended predefined rejection rate
1000 users and 50 providers	0.8
1000 users and 10 providers	Not applicable
100 users and 50 providers	[0.4, 0.6]
100 users and 10 providers	[0.4, 0.6]

Table 2 summarizes the recommended predefined rejection rates for the IPs for the four cases. When the number of users is greater and there is enough providers to serve them all (Case I), we recommend the predefined rate of 0.8. If there is not enough providers to serve them all (Case II), we recommend no predefined rate. On the other hand, if demand is less than the supply, we recommend a predefined rate in the range from 0.4 to 0.6.

VI. CONCLUSIONS

Grid computing allows for large scale resource collaboration. However, seamless collaboration is a challenge due to distributed rules and policies. Agent technology would be suitable to meet this challenge, since agents are autonomous and intelligent. Pricing helps users to access legally distributed resources which belong to different owners. In this work, we develop a multi-agent based continuous double auction model to maximize profit for providers. We presented our results for different cases with different number of users and providers. We showed that our agent-based intelligent providers perform better than traditional providers in most cases. Currently we are working on how provider agent can sense current number of requests autonomously and adjust the predefined rate accordingly.

REFERENCES

[1] Foster, I., and C. Kesselman, *The grid blueprint for a future computing infrastructure,* Morgan Kaufmann, 1999.

[2] T. Zhu, and R. John, "Market-based grid resource allocation using a stable continuous double auction", In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, IEEE Computer Society, 2007.

[3] K. L. Mills, and C. Dabrowski, "Can Economics-based Resource Allocation Prove Effective in a Computation Marketplace?", *Journal of Grid Computing*, 2008, pp. 291–311.

[4] A. Marcos, and R. Buyya, "An evaluation of communication demand of auction protocols in grid environments", In *Proceedings of the 3rd International Workshop on Grid Economics and Business (GECON, 2006)*, World Scientific Press, 2006.

[5] R. Buyya, D. Abramson, and S. Venugopal, "The Grid Economy", In *Proceedings of the IEEE,* 2005, pp. 698-714.

[6] Wooldridge, M. *An Introduction to MultiAgent Systems,* John Wiley & Sons Ltd., Chichester, England, 2002.

[7] W. Marek, P. Stefan, P. Radu, and F. Thomas, "Applying double auctions for scheduling of workflows on the Grid", *Proceedings of the ACM/IEEE conference on Supercomputing*, IEEE Press, Austin, Texas, 2008.

[8] B. Pourebrahimi, K. Bertels, G. M. Kandru, and S. Vassiliadis, "Market-Based Resource Allocation in Grids", *Second IEEE International Conference on e-Science and Grid Computing (e-Science '06),* 2006, pp. 80-80.

[9] Y. Sallez, T. Berger, and C. Tahon, "Simulating intelligent routing in flexible manufacturing systems using NetLogo", *IEEE International Conference on Industrial Technology*, 2004, pp. 1072-1077.

[10] R.-C. Damaceanu, "An agent-based computational study of wealth distribution in function of resource growth interval using NetLogo", *Applied Mathematics and Computation,* 2008, pp. 371-377.

[11] J. N. Michael, T. C. Nicholson, and R. V. Jerry, "Experiences creating three implementations of the repast agent modeling toolkit", *ACM Trans. Model. Comput. Simul.*, 2006, pp. 1-25.