C-024

An Auction based Resource Allocation Considering Multifaceted Utilities in a Peer to Peer Environment

Chainan Satayapiwat* Kazuhiko Komatsu† Ryusuke Egawa† Hiroyuki Takizawa* Hiroaki Kobayashi†

Abstract

Recently, many market-based approaches have been studied as one of the promising alternatives in a resource allocation problem. Especially, auction-based approaches are widely chosen due to its distributed nature and its relatively lower complexity. However, employing an auction to allocate jobs is only suitable for homogeneous environments of resources. This paper proposes an auction-based resource allocation mechanism which enables resource allocation in a heterogeneous environment while minimizing user's inputs. Our preliminary results show that our resource allocation mechanism improves the performance of important jobs during high-loaded.

1 Introduction

Resource allocation is one of key challenges in designing a computing platform over a peer-to-peer environment. Many recent researchers have designed resource allocation mechanisms based on market-based techniques because such mechanisms can encourage users to use resources only when a submitted job is important during high-loaded. Thus, the market-based resource allocation can reduce the turnaround time required for high priority jobs. This is an attractive property for resource sharing in a large-scale distributed peer-to-peer system where users assume to be selfish.

In general, auction protocols for resource allocation consider only a price factor when matching jobs to resources. However, we assume that resources in a peer-to-peer system are heterogeneous. In such an environment, there are two ways to create a market. One is to create one market for each resource as shown in Figure 1-(a). The other is to allow one market to trade multiple resources in Figure 1-(b). In Figure 1-(a), each market trades only resources of the same performance. To handle various resources, there are many markets and this leads to inflexible resource allocation. For example, when a user requires a resource satisfying some requirements, the user may not be able to decide which market can provide such a resource, resulting in bids at all the market. In Figure 1-(b), all the resources are traded at the same market. As a result, the resource offered at the lowest price might be too low performance for the price. In other words, it might be wise to choose the next lowest price offer which provides higher performance when the price is not much different from the lowest one.

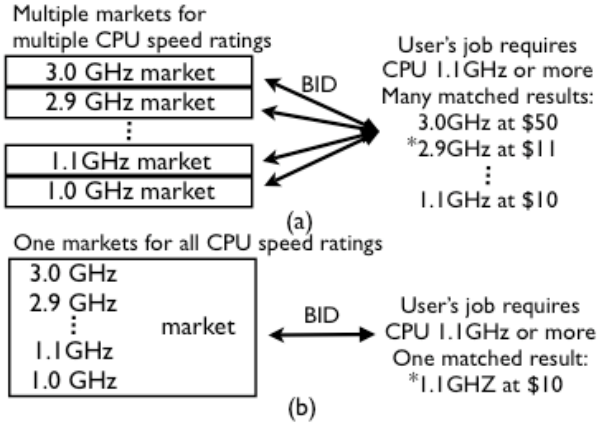

Figure 1: Problems in existing auction models. Star sign indicates the chosen result.

This work aims to overcome these problems by extending our previous work [1], which is designed assuming a Grid environment. This previous work provides a framework to design a resource allocation mechanism, which considers any numbers of criteria in making matching decision. A user uses weighted values to inform the resource allocation mechanism of his preference in the matched result, while a provider uses the weighted values to inform the preference in accepting each job. However, the proposed criteria assume that users know an accurate job processing time and providers can make advance reservation. These assumptions are valid in a Grid environment but not in a peer-to-peer environment.

This work extends the previous work to make it suitable for a peer-to-peer environment by changing the considered criteria so that resource allocation can be free from the unrealistic assumptions in a peer-to-peer environment such as accurate estimation of processing times. Moreover, users should not need to concern about the money management or providing the *utility function*, which represents the relationship between the completion time and user's job valuation. This work minimizes the user's inputs required in submitting a job and this makes the resource allocation transparent from user's standpoint.

2 Related Work

The major disadvantage of existing auction-based resource allocation approaches is that the matching is done only by considering the price. Thus, user's matched resources are guaranteed to be the cheapest ones but not the worthiest.

*Graduate School of Information Sciences, Tohoku University
†Cyberscience Center, Tohoku University

General auction models, including their variations such as SCDA [4], have this drawback.

Several approaches have been introduced to solve this problem. CompuP2P[3] defines a market to trade only one specific resource. If a resource has various speeds such as CPU, which has many different capabilities, one market handles trading of resources in a small range of speed. However, the auction protocol may, again, match with a relatively low performance resource which has the lowest price in this range. In addition, the best capability of the resource that the job can be matched to is the upper limited of resource capability in that market range but the next higher market range may have a better resource offering at a lower price. Moreover, the way to simultaneously deal with multiple criteria has not been established.

Another approach uses a utility function to represent the relationship of completion time and user's job valuation [2]. By calculating the finished time and using the utility function to maximize the user's valuation, the allocation results have achieved a balance between the price and the performance that a user desires. However, this requires an accurately-estimated processing time to calculate the finished time of the job on the resource, even though the estimation is difficult. In addition, a user has to define a utility function, although its definition methodology is not established yet.

3 Auction Protocol

3.1 Proposal Overview

Our approach uses the utility function to make a decision. However, we avoid defining the price relationship with the completion time because this requires knowing the estimated processing time. Instead, our proposal considers resource performance and price in the decision. The final awarding decision is based on the user's *utility value* on each offer. The utility value can be calculated using a utility function. This utility function allows user to express the importance of each criterion as *weighted values*.

This proposal assumes that an auctioneer, which is the market where resources are traded, controls the auction and acts in the middle between multiple users and multiple resource providers. Users send job's weighted values and minimum requirements to an auctioneer to start an auction. Then, the auctioneer opens the auction by informing the received data to every registered resource provider which meets the minimum requirements. Using the provided information, each provider constructs an offer for this job and sends it to the auctioneer. Finally, the auctioneer calculates a utility value for each offer and awards the provider with the highest utility value. This procedure is repeated whenever a new job is arrived at the auctioneer and hence there will be only one job in each round of the auction.

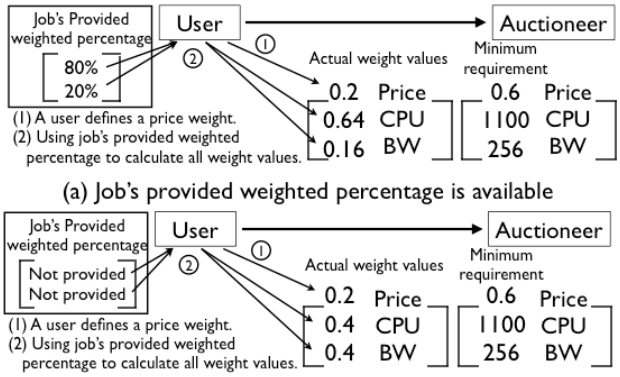

Figure 2: Example of a user's submitted job and weighted values calculation

3.2 Deriving weighted values

In this proposal, users consider three criteria which are (1) a price criterion, (2) CPU criterion (available processing cycles), and (3) BW criterion (an available network bandwidth). However, other criteria, such as memory size, can be introduced in a similar manner. Users start an auction by providing job's minimum requirements and job's associated weighted values of these three criteria. Although it is possible to leave the user to define these three weighted values, this is not preferable. Hence, this proposal offers the ways to eliminate a part of user's inputs as follow.

1) Eliminating the price weighted value input: This proposal introduces an algorithm to automatically decide the price weighted value. This algorithm works by increasing the price weighted value to improve performance when a user has money over a threshold, while decreasing the price weighted value to minimize the expense when money is lower than the threshold. However, it allows users to manipulate the price weighted value by specified job as *priority*. This informs the system to overwrite the price weighted value and maximum budget by not to consider the price factor.

2) Eliminating the CPU and BW weighted value inputs: Each job should have its metadata which specifies *job's provided weighted percentages*. These values are the percentage of actual weighted values, which directly influence the resource allocation mechanism to offer resources in the proportion of these values. It should be noted that job's provided weighted percentages are not mandatory but accurate values result in better proportion of resource performances allocated to the job. If the job's provided weighted percentages are not available for a job, the resource allocation mechanism automatically assigns weighted values by spreading equally over all criteria. Figure 2 summarizes the user-defined information.

3.3 Constructing an Offer to an Auctioneer

After the auctioneer received the job requirements and the weighted values from a user, it opens the auction

by forwarding the job's minimum requirements and job's weighted values to every registered resource provider. Then, each provider has to construct the offer for this job and sends it to the auctioneer.

Firstly, we propose a simple method to construct offers as shown in Equation (1). Min_i is the minimum requirement on a criterion i. C_i is a user's criterion weighted value in criterion i. k is an *offer's resource modifier* which increases additional proportion of resource performances.

$$Offer_i = Min_i + \frac{Min_i C_i k}{\sum_{j=1}^{2} C_j}. \quad (1)$$

For each offer, *a based price* is the price that is associated to each resource offer. This is calculated based on the load percentage of each resource with assumption that this job is admitted. The based price increases slowly when the load is low, while it increases aggressively when the load becomes higher to avoid the resource providers from being overload. Equation (2) shows the price calculation where $Price_i$ is the based price in a criterion i. Providers are free to decide its minimum price α and its weight on each factor β_i to adjust the price as needed. The result price is the price that will charge the user over the usage time.

$$Price = \alpha + \sum_{i=1}^{2} Price_i \beta_i. \quad (2)$$

To sum up, the resource provider calculates offers with different k values to generate offer candidates and the price of each candidate using Equations (1) and (2), respectively. Then, it calculates utility values for each offer to select the best offer among these candidates. Finally, the offer with the highest utility value is returned to the auctioneer. If there is no populated offer satisfying the minimum requirement, the provider does not need to reply to the auctioneer.

3.4 Matching at the Auctioneer

After all resource providers submit their offers to the auctioneer, the auctioneer applies the user's weighted values, the offer's price, and the offer's resource performance to the utility function (UF) to find the utility values. The utility function is defined as follows. Given C_i is a user's criterion weighted value in criterion i and P_{ij} is a performance perceived by a user's job on criterion i when allocate to provider j.

$$UF(C_1, C_2, ..., C_n, P_{1j}, P_{2j}, ..., P_{nj}) = \sum_{i=1}^{n} C_i P_{ij}. \quad (3)$$

P_{ij}, which ranges from 0 to 1, is modeled linearly in this work. Value 0 represents an offer promising resources equal to the minimum requirement and value 1 represents the best offer performance, respectively.

Table 1: Nodes in the system

NodeID	CPU rating (MIPS)	Bandwidth rating (MB/Sec)
0	1500	25
1	2000	100
2	2000	100
3	2500	50
4	3000	50

Table 2: Workload setup (jobSize in million instructions, network Datasize in MB, mCPU in MIPS, and mBW in MB per seconds)

Type	jobSize	network DataSize	mCPU	mBW	CPUw/ BWw
1	49500	720	1100	16	0.8/0.2
2	32985	990	733	22	0.5/0.5
3	19800	585	440	13	0.5/0.5
4	24750	1485	550	33	0.2/0.8

Finally, the provider with the highest utility value is awarded. The auctioneer informs the result to both job's user and the auction's winner about the result and the user may start submitting the job for execution.

4 Performance Evaluation

4.1 Experimental Setup

To evaluate the effects of the proposed mechanism, this paper evaluates the allocated resources' performance offered to the user for each set of weighted values. In addition, this paper also evaluates the total turnaround time of jobs and compares it to a standard round robin scheduler, which assigns a job to the first resource provider that satisfies job's minimum requirements in the circular order. As the market-based strategy, it should have an ability to improve the turnaround time of the high-priority jobs during the high load. Thus, this paper simulates a high-loaded period and gives high priority to 50% of jobs. Then, the turnaround time of priority jobs is measured and compared to the round robin scheduler.

Node's setups used in this simulation are shown in Table 1. The CPU performance is represented by the number of million instructions per seconds and network bandwidth is measured by the data size transferred per second. All nodes are assumed to have the same architecture and only differ in their CPU performance. Besides, this simulation assumes that the communication does not occur in parallel with CPU computation and there is no communication among nodes listed in Table 1. Thus, we calculate the total processing time of node i using Equation (4).

$$Time = \frac{jobSize}{CPURating} + \frac{networkDataSize}{bandwidthRating}. \quad (4)$$

When creating an offer, the offer's resource modifiers are 0, 1, and 3 to populate offers with approximately 1, 2, and 4 times higher than original minimum requirement in each criterion. To calculate the price, we assume the minimum price α is 0 and pricing factor's weight β is 1 for every factor. This means that the provider values the importance to all resource equally.

A workload contains 40 jobs with four types of jobs as shown in Table 2. mCPU and mBW are the minimum requirements, and CPUw and BWw are job's provided weighted percentages associated to each job type. Each job type is generated evenly throughout the simulation with 10 seconds time interval between each job. To examine the priority job's behavior in a high-loaded period, we reduce this interval to 2 seconds per job during last 20 jobs and assign priority to even number jobs.

4.2 Experimental result and discussion

Figure 3 shows the average resource performance allocated to each job type, compared to its minimum requirement. These results show that weighted percentage values can provide a good approximation of the resource performance requested by a user. This means that these percentages can be adjusted to achieve the anticipated performance gains when additional resources are available. In addition, the jobs with lower requirements tend to have higher percentage of resources because it is easier for the resource provider to commit and allocate an offer with a high offer's resource modifier.

Table 3 shows the average turnaround time. The results show that our scheduler performs slightly worse than the round robin scheduler only if we average the turnaround times for all jobs from the beginning to the end of simulation. However, the strength of the market-based resource scheduler is its ability to limit a low priority job from entering the system. This provides more available resources to jobs with high priority. During the highly loaded period, the priority jobs achieve around 20% improvement compared to the round robin strategy. This improvement is because the low priority jobs have a lower budget threshold and higher price weighted value. This reflects that the users do not value these jobs and they are not worth to execute at the high load time and hence the priority jobs gains more available resources.

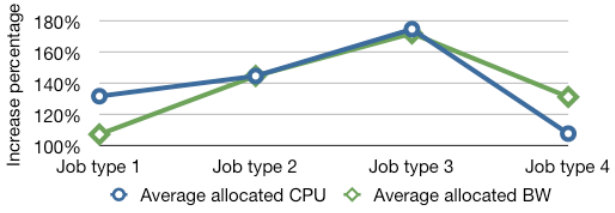

Figure 3: Effect of weighted values to allocated resource performance for each job type

Table 3: Average turnaround time (seconds)

Type	Round robin	Our scheduler
All time / All jobs	46.73	48.80
High load / All jobs	61.35	68.90
High load / Pri. jobs	53.60	42.20

5 Conclusions and Future Works

This paper has proposed a new auction-based resource allocation mechanism for peer-to-peer environments. Instead of matching resources to jobs regarding the price, we include the resource performance of the offers in making decision. In this way, our proposal allows a user's jobs to obtain a matched resource which is both low price and high performance. Furthermore, this proposal eliminates the need of estimated processing time which is difficult to be accurately predicted. The experimental results show its ability to handle high priority jobs in a highly-loaded system. In addition, the experimental results also indicate that the allocated resource performance is closely related to the weighted value given. As a future work, we will further compare the mechanism with other market-based schedulers.

References

[1] C. Satayapiwat, R. Egawa, H.Takizawa, H. Kobayashi. A Utility-Based Double Auction Mechanism for Efficient Grid Resource Allocation. IEEE International Symposium on Parallel and Distributed Processing with Applications, pages.252-260, 2008.

[2] David E. Irwin, Laura E. Grit, and Jeffrey S. Chase. Balancing risk and reward in a market-based task service. Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing, pages 160-169, 2004.

[3] R. Gupta and A. Somani. Compup2p: An architecture for sharing of computing resources in peer-to-peer networks with selfish nodes. Proceedings of the Second Workshop on the Economics of Peer-to-Peer Systems, 2004.

[4] Z. Tan and J. R. Gurd. Market-based grid resource allocation using a stable continuous double auction. Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, pages 283-290, 2007.