A Distributed Market Framework for Large-scale Resource Sharing *

Marian Mihailescu and Yong Meng Teo

Department of Computer Science, National University of Singapore
Computing 1, 13 Computing Drive, Singapore 117417
[marianmi,teoym]@comp.nus.edu.sg

Abstract. Current distributed computing infrastructures, such as peer-to-peer networks, grids, and more recently clouds, make sharing and trading resources ubiquitous. In these large distributed systems, rational users are both providers and consumers of resources. Currently, there is growing interest in exploiting economic models for the allocation of shared computing resources that incentivize rational users. However, when the number of resource types and users increases, computational complexity of the allocation algorithms grows rapidly and efficiency deteriorates. In this paper, we propose a scalable distributed market framework for the allocation of shared resources in large distributed systems. We use mechanism design to create a pricing scheme that allocates a request for multiple resource types, by trading economic efficiency for computational efficiency, strategy-proof and budget-balance. To address scalability, our proposed framework leverages on a peer-to-peer overlay for resource discovery and management. We prototype our framework using FreePastry, a popular overlay network based on the Pastry protocol. We show that our scheme is efficient and scalable using both simulation experiments and results from the deployment on PlantLab.

1 Introduction

Current distributed systems are slowly converging towards federated sharing of computing resources [7, 11]. In peer-to-peer networks, grid computing and more recently cloud computing, users share and trade different types of resources over the network [22–24]. Recent results show that users that share resources are rational, i.e. they create strategies and manipulate the system in order to achieve their own objectives and maximize their own benefit [8, 15]. For example, performance in file-sharing peer-to-peer networks is affected by free-riders, users that consume more than their fair share [19]; in a computational grid, users compete for the same resources, which results in increased waiting times [22]; some users of SETI@home, a popular distributed computing project, modified the software client to report false-negatives in order to achieve higher rankings [19].

Market-oriented economies have been successful in managing resource allocation in social systems with rational users. Consequently, there is growing interest

* This is a revised version of the paper published in the Proceedings of Euro-Par, pp. 418-430, LNCS 6271, Springer-Verlag, Ischia, Italy, Aug 31 - Sept 3, 2010.

in adopting market-based approaches for the allocation of shared resources in computational systems. In addition to rational users, an economic-inspired resource allocation system can also address the dynamic nature of demand and supply. This is useful in systems where resource demand and supply can vary significantly over time, such as flash crowds or distributed systems with unreliable hosts that are geographically distributed over different administrative domains.

In this paper we study the *scalability* of dynamic market-based sharing systems when the number of market participants and the number of resource types in a request increases. Specifically, pricing mechanisms proposed for the allocation of shared resources are either computationally hard, such as combinatorial auctions [5], or inherently centralized, such as double auctions [6]. Thus, increasing the number of peers or the number of resource types in a request leads to large waiting times for users, due to the computational complexity and the communication complexity, respectively. Consequently, we propose a *distributed market framework* where rational users are both buyers and sellers of resources. Our approach is to leverage existing work in peer-to-peer systems and use a distributed hash table (DHT) to manage resource information. Specifically, we map a user *buy* request on the DHT *lookup* operation, and a user *sell* offer on the DHT *store* operation. For pricing, our framework makes use of a reverse auction-based mechanism that is able to allocate a request containing more than one resource types [21]. In addition to multiple resource type allocations, other properties achieved by the pricing mechanism are incentive compatibility and budget balance, at the expense of economic efficiency. However, according to the Myerson-Satterthwite impossibility theorem [14], no pricing mechanism can be efficient, strategy-proof, and budget-balanced at the same time.

The paper is organized as follows. In Sec. 2 we discuss related work and the main issues in distributed resource pricing: economic properties, such as strategy-proof and Pareto efficiency; computational complexity; and scalability. Next, in Sec. 3, we propose a distributed market framework built on top of a peer-to-peer overlay, where all peers participate in allocating resources, as buyers and sellers, or determine an allocation, as request brokers or resource brokers. To determine the allocation time we perform theoretical and experimental analysis in Sec. 4. Lastly, Sec. 5 presents our conclusions and insights for future work.

2 Related Work

Market-based solutions for the allocation of shared resources have been previously proposed, but need to improve either the economic or the computational efficiency in order to become practical. Bellagio and Mirage [2] are two market-based allocation systems that focus on maximizing economic efficiency. Accordingly, both systems use a repeated, sealed-bid combinatorial auction [5], a pricing mechanism that is Pareto-efficient and strategy-proof. Moreover, in a combinatorial auction the users can bid on bundles of resources, containing more than one resource types, as opposed to individual resources. However, determining which users get allocated using a combinatorial auction is a NP-hard problem [17]. Thus, both Bellagio and Mirage use a greedy algorithm to determine the allocation winners, such that the allocation time does not depend on the num-

ber of resource types in the buyer request [4]. There are two drawbacks in this approach. Firstly, using a greedy algorithm leads to an outcome that is not Pareto-efficient. Secondly, the allocation mechanism is no longer strategy-proof. Thus, economic efficiency is further reduced when users are untruthful [4]. The distributed market framework we propose in this paper uses a strategy-proof pricing mechanism [21]. Although Pareto efficiency is not achieved, economic efficiency is not affected by the users' degree of untruthfulness.

Several market-based allocation systems for grids, such as Sorma [16] and Nimrod/G [3], use bargaining or negotiation to determine the resource price. The advantage of this approach is that sellers and buyers communicate directly, without a third party mediating an allocation. However, bargaining results in high communication costs. In a large dynamic market, each buyer has to negotiate with all sellers of a resource type in order to maximize his utility. The communication costs further increase when a buyer requires more than one resource types. Thus, scalability is an issue both when increasing the number of users and the number of resource types in a request. We propose to manage resource information using a peer-to-peer overlay network, where each resource type lookup can be processed in parallel by different peers.

Cloud computing uses fixed pricing to provide computational resources and services on-demand over the Internet. Although simple to implement, fixed pricing is not suitable for a system with multiple providers, or where users are both sellers and buyers of resources [13]. Federated clouds, a topic of recent interest, aims to integrate cloud resources from different providers, to increase scalability and reliability [4]. With a large number of providers (sellers) and users (buyers), fixed pricing cannot adapt to the changes in demand and supply. More suitable for federated clouds, dynamic pricing mechanisms such as the one used in the proposed framework sets resource payments according to demand and supply.

In PeerMart [10], the authors propose a distributed market for peer-to-peer services built on top of a peer-to-peer overlay. Although resource location in PeerMart and the proposed framework is similar, our framework provides several key advantages. Firstly, PeerMart does not support multiple resource type allocations. When a user requires several resource types, it has to aggregate resources manually, which is not efficient. Secondly, PeerMart is not strategy-proof. Pricing takes place using a simple double-auction mechanism, where the payment is the mean price between seller price and buyer price. Thus, users are encouraged to submit untruthful prices to increase their utility.

3 Proposed Distributed Market Architecture

We have identified three major components that constitute the market architecture: resource location, the pricing mechanism, and allocation administration.

A fundamental problem in dealing with large collections of shared resources is resource location. As shown in Fig. 2, the resource market receives resource information from sellers in *publish* messages, and query requests from buyers in *lookup* messages. The time required to perform resource lookup is a significant part in the total allocation time. Searching for available resources in a lookup takes longer time when the number of resources is large. Moreover, resource

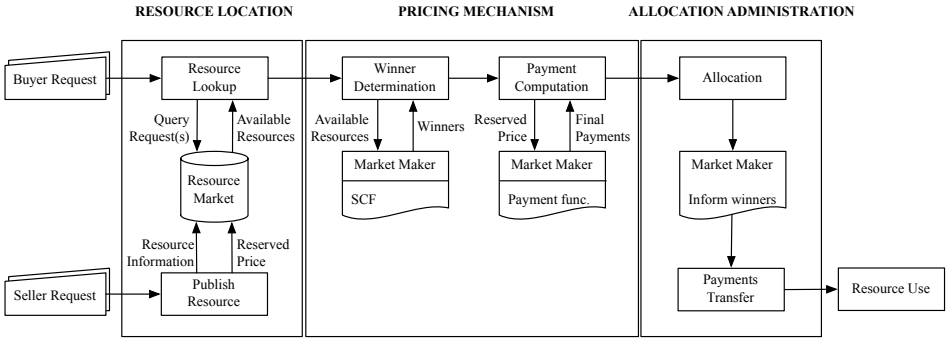


Fig. 1. Market Architecture

information and availability can vary over time, as resources join, leave, or fail in the system. To perform efficiently in these conditions, the resource location service requires scalability and support for dynamic resources.

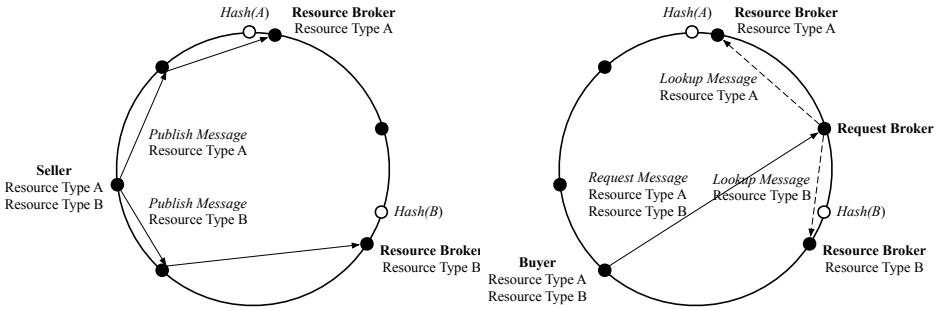
The pricing mechanism is key part in the resource allocation framework. Based on the social choice function (SCF) and available resources, the pricing mechanism determines which users are allocated (Winner Determination). Next, using the published prices and the selected payment functions, the pricing mechanism computes the user payments, both for sellers and buyers (Payment Computation). The performance of the pricing mechanism performance takes into consideration several features: strategy-proof, budget-balance, Pareto-efficiency, and multiple type allocations; the computational efficiency; and scalability. Additionally, the performance of the pricing mechanism is measured by the percent of successful requests and resources allocated.

In our previous work [21], we describe the resource market using a mechanism design problem and propose a reverse auction-based pricing mechanism with Vickrey-Clarke-Groves [9] seller payments and the sum of seller payments for the buyer. The proposed pricing mechanism allocates resources for single or multiple resource types buyer requests, while achieving incentive-compatibility and budget-balance. For simplicity, we have considered a centralized market-maker that manages the available resources and requests. However, our results have shown that scalability becomes an issue when the number of users or the number of resource types is increased [13]. In this paper, we focus on resource location and propose a scalable, distributed market framework. We refer to *vertical scalability* when increasing the number of resource types, and *horizontal scalability* when increasing the number of users.

In the last step of resource allocation, the winning sellers and buyers are informed of the allocation, and the buyer can start using the resources. In addition, payments take place. Additional features such as management or monitoring, can be added to allocation administration. The performance of allocation administration depends on the payment system used by the framework, which must be scalable and secure.

3.1 Distributed Auction Mechanism

In order to maintain the economic properties of the proposed pricing mechanism, the distributed payment computation requires complete information about re-



(a) Seller and Resource Brokers (b) Buyer and Request Broker

Fig. 2. Distributed Resource Market

sources in the system in order to determine payments. In our distributed model, resource information such as owner address, resource type, number of items, and the cost for each item, is divided between different hosts to provide scalability. Similarly, a buyer request is also divided into several lookups that are sent to the relevant hosts for the price to be computed. Accordingly, our approach is to use distributed hash tables (DHT), as the infrastructure to maintain resource information and to build a scalable resource location service for buyers, where resource information is distributed according to the resource type. Peers in a DHT are organized as a structured overlay network, and data is mapped to nodes in the overlay network based on their identifier. A DHT provides an interface *lookup(key)* for retrieving a key-value pair. The key is an identifier assigned to a resource, while the value is an object from the DHT. To facilitate object location, a DHT overlay network provides a routing operation, based on the destination node key. Thus, the distributed hash table is a natural data structure to support the proposed distributed resource market model, where the resource type is the key used to store the list of sellers for the respective resource type.

Figure 2 shows the basic architecture of our system. A *peer* represents a user that joins the distributed market overlay. All peers can perform any of the following roles:

Seller: A peer becomes a seller after publishing a resource. The seller sends a different publish message for each resource type, containing the number of available items and the cost for each item. The publish operation is performed using the DHT store interface, using the hash of the resource type as the *key*. Thus, the published resource information is stored by the *resource broker*, the peer with the identifier closest to the hash of the resource type.

Buyer: A peer becomes a buyer when it sends a request message. The request message contains one or more resource types, the number of items for each resource type, and the total price the buyer is willing to pay. The *key* for the request message is a random identifier. Accordingly, the request message is routed by the overlay to the peer with the identifier closest to the random key, which becomes a *request broker*.

Resource Broker: A resource broker is the peer in the overlay network with the identifier closest to the hash of a resource type published by sellers. The relation "closest" is defined by the specific overlay implementation, e.g. nu-

merically closest forPastry [18], the first node in clockwise direction for Chord [20], etc. Resource brokers keep a list of published resources for each resource type they are responsible for. After receiving a lookup request, the resource broker determines the winners and computes the payments for the respective resource type. If the allocation is successful, it updates the resource list and informs the winning sellers about the allocation and payments.

Request Broker: A peer in the overlay becomes a request broker when its node identifier is closest to a buyer request identifier. When receiving a buyer request, the broker sends a lookup for each resource type in the request, using the hash of the resource type as the *key*. Thus, each lookup is routed by the overlay to the resource broker responsible for the respective resource type. After receiving seller payments for each resource type, the request broker computes the buyer payment. If the allocation is possible, i.e. the buyer payment is less or equal to the buyer price, the request broker sends a commit message directly to the resource brokers. In addition, the request broker informs the buyer about the payments and allocation.

3.2 Deadlock-free Auctions Protocol

The distributed auction mechanism outlined above is, in fact, a two-phase commit protocol. Firstly, a buyer request is routed by the overlay to a request broker. Then, for each resource type, the request broker sends a lookup message using the hash of the resource type as the *key*. This corresponds to the *commit-request* phase. The lookup messages are routed by the overlay network to the resource brokers responsible for the respective resource types. Next, the resource brokers compute payments and reply to the request brokers, similar to an *agreement* message. Lastly, the request brokers send the *commit* message.

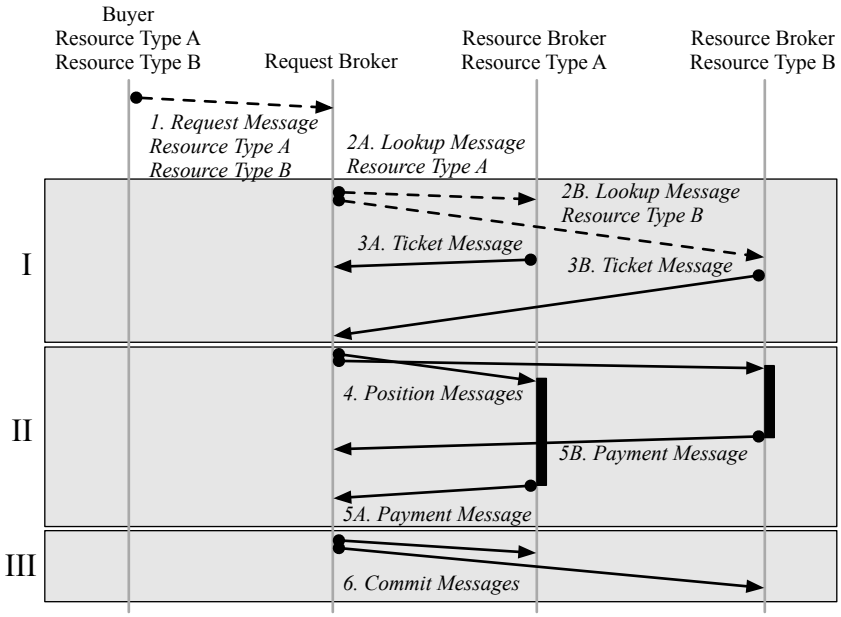


Fig. 3. Proposed Auction Protocol

Using a blocking protocol such as two-phase commit has several implications. Firstly, lookup requests are serialized by resource brokers. Thus, between sending the payment and receiving the commit message, resource brokers are not able to compute payments for other requests. However, this strategy helps maintaining the strategy-proof property of the pricing mechanism we employ. In [21], we show that buyer strategy-proof is achieved by selecting requests independent of the buyer valuation, such as the first-come-first-serve strategy. Accordingly, a blocking protocol implements the same requirement for a distributed market. One of the drawbacks of using a two-phase commit is that concurrent requests for more than one resource types may lead to deadlocks, when lookup messages for different requests arrive at multiple resource brokers in different order.

In order to prevent deadlock, we propose an algorithm which employs a three-phase commit protocol in conjunction with a synchronization mechanism inspired from Lamport's logical clocks [12]. Figure 3 contains a diagram of the deadlock-free auction protocol. Initially, the buyer request is routed by the overlay network to the request broker. We use dotted arrows for routed messages and solid arrows for direct peer-to-peer messages. In the first phase of the proposed protocol (I), the request broker sends a lookup message for each resource type in the buyer request. Each resource broker responds with a *ticket* message, which contains the number of the next request in the resource broker lookup queue. The ticket numbers are used for synchronization in the second phase similar to the Lamport logical timestamps.

The request broker waits for all ticket numbers corresponding to each resource type in the buyer request. In the second phase (II), after all ticket numbers are received, the request broker sends to all resource brokers a *position* message containing the maximum ticket number. All resource brokers re-order their lookup queues by moving the respective lookup from the *ticket* to the *position* location in the queue. Next, the resource brokers compute payments for the lookup message in the head of the queue.

In the last phase (III), after the request broker receives all payments, the *commit* message is sent with the result of the allocation. If the allocation is successful, the buyer and sellers are sent allocation and payment information.

The advantage of the proposed distributed auction mechanism is twofold. Firstly, buyer requests for different resource types are processed in parallel. In contrast to the centralized model, where all requests are serviced by the market-maker sequentially, having a distributed market allows different resource brokers to make concurrent allocations. Secondly, the payment computation for the same request is also parallelized. Thus, the request broker distributes a request using several lookup messages, one for each resource type. Each lookup is processed by different resource brokers, and the computation time for the allocation is reduced to the computation time for one resource type. Using the peer-to-peer model, where any peer can be a seller, buyer, resource broker and request broker, adds scalability to the our market-based resource allocation framework.

4 Analysis

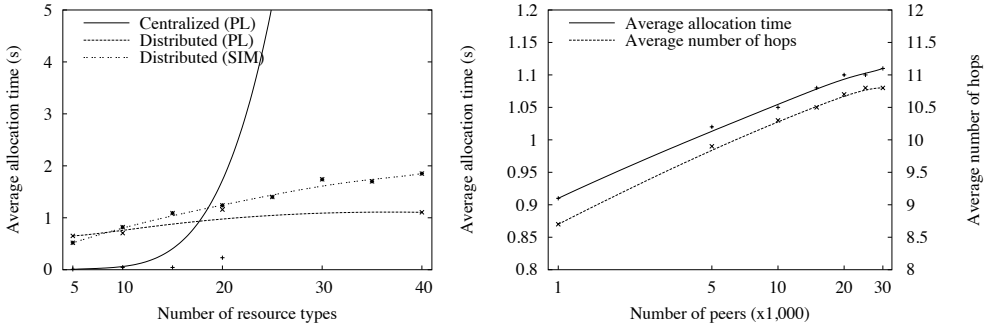
We analyze the vertical and horizontal scalability of the proposed pricing mechanism both theoretical and experimental. We developed a prototype implementation of the proposed distributed market using FreePastry [1] as the underlying overlay network. FreePastry allows us to measure the performance of our pricing and allocation scheme in two different environments: the FreePastry simulator, where we can simulate a large distributed market, and PlanetLab, where we deployed our framework and use it to validate the simulator results on a smaller number of distributed nodes. In the theoretical analysis, we identify the factors that affect the request allocation time. We verify the result of the theoretical analysis against experimental results obtained using simulations.

4.1 Vertical Scalability

To study vertical scalability, with respect to the number of resource types in a buyer request, we compare the average allocation time in the distributed market implementation with a centralized implementation. In the latter, we delegate one peer as the market-maker, while the other peers are either buyers and sellers.

As shown in Fig. 4(a), we use 5, 10, 20 and 40 resource types in a buyer request for the centralized and distributed implementation, both on PlanetLab and the simulator. We create an overlay network containing 50 peers, where each peer generates 100 events. The total interarrival rate of the events is exponentially distributed with the mean of one second. We consider a balanced market, where demand equals supply. Accordingly, each peer event has an equal probability for a publish or a request message. Additionally, the number of items for each resource type is generated from an exponential distribution with mean 10, and the price for a resource item is uniformly distributed between 80 and 120.

Our results show that in the distributed market, the auction protocol imposes a greater overhead and, when having a small number of resource types in a request, the average allocation time is higher than the centralized implementation. However, as the number of resource types increases, the distributed scheme proves to be scalable and maintains a consistent allocation time of less than one second. The average allocation time obtained in simulations is higher than the measured time on PlanetLab because the simulations used a fixed network delay between peers of 100 ms, much higher than the delay between PlanetLab nodes.



(a) Varying the Number of Resource Types (b) Varying the Number of Peers

Fig. 4. Horizontal and Vertical Scalability

4.2 Horizontal Scalability

To study horizontal scalability, we use the FreePastry simulator, as PlanetLab is not able to provide the necessary number of nodes for a scalability study. In our simulations, we were able to create an overlay network of approx. 33,000 nodes before the average peer join time increase exponentially. Consequently, our results summarized in Fig. 4(b) are for network sizes of 1,000, 5,000, 10,000, 20,000 and 30,000 peers. We have used similar settings as in the previous study: one second event interarrival rate, balanced market, number of items from an exponential distribution with mean 10, price uniformly distributed between 80 and 120, and 100 ms fixed network delay. The number of resource types is sampled from an uniform distribution between 1 and 10. We run the simulations for 600,000 events, or approx. 7 simulation days.

Our results show that the proposed distributed market framework is horizontally scalable as the average allocation time increases logarithmic with the number of users. Additionally, we have measured the average number of hops for all messages required by the proposed protocol and found that both curves have the same gradient. Thus, our scheme adopts the horizontal scalability of the underlying overlay network.

4.3 Average Allocation Time

We consider the average allocation time (T_{alloc}) as the total time taken since a buyer sends the request until it receives the allocation results, averaged for all successful buyer requests. We identify three components that determine the total allocation time: i) *communication time* (T_n), which represents the time taken to transmit messages in the network; ii) *queue time* (T_q), which is the time spent by a lookup message in the resource broker queue since the position message is received until payments are computed; and iii) *computational time* (T_c), which is the time taken to run the pricing algorithm:

$$T_{alloc} = T_n + T_q + T_c$$

Communication Time. The communication cost incurred by the allocation of a buyer request is given by the messages exchanged between the buyer, request broker, and resource broker: request message, lookup message, ticket message, position message, payments message, commit message. We consider a stable overlay network, where routing takes at most $\log N$ steps, where N is total number of peers in the overlay. Thus, the request message is routed from the buyer to the request broker in at most $\log N$ hops. Similarly, the lookup message is routed from the request broker to the resource broker in at most $\log N$ steps. After the resource broker receives the lookup message, it can look at the sender address and reply with the ticket number in one hop. Similarly, the position, payments and allocation messages are forwarded in one hop. Considering an average network delay time, d, the total communication time is:

$$T_n = d(\log N + \log N + 1 + 1 + 1 + 1) = 2d(\log N + 2) \quad (1)$$

Service Time. To determine the queue time and the computational time, we assume Poisson arrivals for the buyer requests and we use queuing theory, considering the resource broker a M/M/1 system. The service time is computed as:

$$T_s = T_q + T_c = \frac{1}{\mu - \lambda}$$

where μ is the average service rate of the resource broker, and λ is the average lookup arrival rate (for a resource type). Since the resource broker serializes the requests, the service rate is given by the time since the computation starts, until the commit message is received, when a new lookup can be processed. Accordingly, the service time includes the computation time (T_c), sending the payment message (d), and receiving the commit message (d). Thus, the service rate is:

$$\mu = \frac{1}{T_c + 2d}$$

Accordingly, the total service time is:

$$T_s = \frac{T_c + 2d}{1 - (T_c + 2d)\lambda}$$

For simplicity, we consider that the computation time T_c negligible, compared to the average network delay time d, which in the Internet is ranging from several milliseconds to several hundreds of milliseconds. Thus, ignoring the computation time, we can express the total allocation time for a buyer request as:

$$T_{alloc} = 2d(\log N + 2) + \frac{2d}{1 - 2d\lambda} - 2d = 2d(\log N + 1 + \frac{1}{1 - 2d\lambda}) \qquad (2)$$

In summary, the factors which affect the scalability of the proposed framework are: i) N, the size of the overlay network; ii) d, the network delay; and iii) λ, the arrival rate of requests for a resource type. The previous results (Section 4.2) show that the average allocation time increases logarithmically with N, the number of peers in the network.

4.4 Network Delay

Our theoretical analysis show that network delay is one of the factors that influences the scalability of the proposed distributed market framework. To evaluate the impact of network delay, we have simulated an overlay network of 10,000 peers, with an event interarrival rate of one second in a balanced market. We simulated 600,000 events for different fixed values of network delay, ranging from 100 ms to 900 ms.

For comparison, we plotted average allocation time for the simulations together with the theoretical worst-case scenario in Fig. 5. Although the network delay from the PlanetLab nodes was much lower ($\sim$ 40 ms) than the values used in our simulation, we have found that a large network delay can result in increased user waiting times.

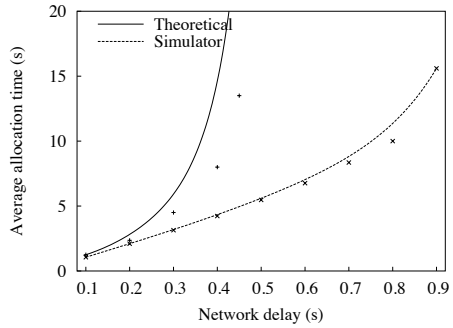


Fig. 5. Varying the Network Delay

5 Conclusions and Future Work

Recent work in distributed systems employs economic models to incentivize selfish users to share resources and to behave according to the rules defined by the system. In this paper, we have considered a market-based approach to resource allocation, where financial incentives are used together with user payments to manage rational users. By having a finite amount of currency to spend on resources, rational users are motivated to spend it based on their expected utility.

In large resource markets, scalability is an issue when information is centralized. We have addressed this issue with a distributed auction scheme, in which resource information is divided according to resource type. Our distributed market leverages on a peer-to-peer overlay to create a resource location service suitable for large markets. Each peer in the overlay network can be a seller, buyer, resource broker or request broker. Resource brokers maintain the resource information and compute payments for each resource type. Request brokers ensure buyer incentive compatibility by implementing a first-come-first-serve strategy for requests having common resource types, while different resource types can be allocated in parallel. Request brokers use the DHT lookup interface to locate the resource brokers for each resource type and decide if allocation is possible. Our results show that for more than 20 resource types, the average allocation time obtained using distributed auctions is around one second, much lower than using a centralized pricing scheme. When increasing the number of market participants, the average allocation time is increased logarithmic.

Our experiments have been performed using a simulator using FreePastry as the underlying overlay network, which was validated by a prototype implementation on PlanetLab. In addition, we performed a theoretical analysis to reveal the remaining factors that influence the scalability of our scheme: large network delays and high request arrival rate. To further increase scalability, the strategic conditions imposed by the pricing scheme can be relaxed. Specifically, in order to achieve incentive compatibility, the current pricing scheme require complete information about resources of the same type. In our future work we intend to investigate resource allocation with incomplete information and the impact of maintaining incentive compatibility has on the overall allocation performance.

Acknowledgments

This work is supported by the Singapore Ministry of Education under AcRF grant number R-252-000-339-112.

References

1. FreePastry - A scalable, decentralized, self-organizing and fault-tolerant substrate for peer-to-peer applications. http://freepastry.org, 2009.
2. A. Auyoung, B. Chun, A. Snoeren, and A. Vahdat. Resource Allocation in Federated Distributed Computing Infrastructures. In *Proc. of the Workshop on Operating System and Arch. Support for the On-demand IT Infr.*, Boston, USA, 2004.
3. R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid. In *Proc. of the 4th Intl. Conf. on High Performance Computing in Asia-Pacific Region*, pp. 283–289, Beijing, China, 2000.

4. R. Buyya, and K. Bubendorfer (editors). Market Oriented Grid and Utility Computing. Wiley Press, 2009.
5. P. Cramton, Y. Shoham, and R. Steinberg (editors). Combinatorial Auctions. MIT Press, 2006.
6. T. Eymann, M. Reinicke, O. Ardaiz, P. Artigas, L. D. de Cerio, F. Freitag, R. Messeguer, L. Navarro, D. Royo, and K. Sanjeevan. Decentralized vs. centralized economic coordination of resource allocation in grids. In *European Across Grids Conf.*, pp. 9–16, Santiago de Compostela, Spain, 2003.
7. J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the Cost of Multicast Transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001.
8. B. Gon Chun, R. Fonseca, I. Stoica, and J. Kubiatowicz. Characterizing Selfishly Constructed Overlay Routing Networks. In *Proc. of INFOCOM 2004*, pp. 1329–1339, Hong Kong, China, 2004.
9. T. Groves. Incentives in Teams. *Econometrica*, 41(4):617–631, 1973.
10. D. Hausheer. PeerMart: The Technology for a Distributed Auction-based Market for Peer-to-Peer Services. In *Proc. of the 40th IEEE Intl. Conf. on Communications*, Seoul, Korea, 2005.
11. K. Krauter, R. Buyya, and M. Maheswaran. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. *Software Practice and Experience*, 32:135–164, 2002.
12. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.
13. M. Mihailescu and Y. M. Teo. Strategic-Proof Dynamic Resource Pricing of Multiple Resource Types on Federated Clouds. In *Proc. of the 10th Intl. Conf. on Algorithms and Architectures for Parallel Processing*, pp. 337–350, Busan, Korea, 2010.
14. R. B. Myerson and M. A. Satterthwaite. Efficient Mechanisms for Bilateral Trading. *Journal of Economic Theory*, 29(2):265–281, 1983.
15. S. J. Nielson and S. A. Crosby. A Taxonomy of Rational Attacks. In *Proc. of the 4th Intl. Workshop on Peer-to-Peer Systems*, pp. 36–46, Ithaca, USA, 2005.
16. J. Nimis, A. Anandasivam, N. Borissov, G. Smith, D. Neumann, N. Wirström, E. Rosenberg, and M. Villa. SORMA - Business Cases for an Open Grid Market. In *Grid Economics and Business Models*, pp. 173–184, Berlin, Germany, 2008.
17. N. Nisan. Bidding and Allocation in Combinatorial Auctions. In *Proc. of the 2nd ACM Conf. on Electronic Commerce*, pp. 1–12, Minneapolis, USA, 2000.
18. A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Address, and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. of the Intl. Conf. on Distributed Systems Platforms*, pp. 329–350, Heidelberg, Germany, 2001.
19. J. Shneidman and D. C. Parkes. Rationality and Self-Interest in Peer to Peer Networks. In *Proc. of the 2nd Intl. Workshop on Peer-to-Peer Systems*, pp. 139–148, Berkely, USA, 2003.
20. I. Stoica, R. Morris, D. Liben-Nowell, D. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *Networking, IEEE/ACM Transactions*, 11(1):17–32, 2003.
21. Y. M. Teo and M. Mihailescu. A Strategic-proof Pricing Scheme for Multiple Resource Type Allocations. In *Proc. of 38th Intl. Conf. on Parallel Processing*, pp. 172-179, Vienna, Austria, 2009.
22. R. Wolski, J. S. Plank, J. Brevik, and T. Bryan. G-commerce: Market Formulations Controlling Resource Allocation on the Computational Grid. In *Proc. of the Intl. Parallel and Distributed Processing Symp.*, pp. 46–54, San Francisco, USA, 2001.
23. C. Wu, B. Li, and Z. Li. Dynamic Bandwidth Auctions in Multioverlay P2P Streaming with Network Coding. *IEEE Transactions on Parallel Distributed Systems*, 19:806–820, 2008.
24. C. S. Yeo and R. Buyya. A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing. *Software: Practice and Experience*, 36:1381–1419, 2006.