Best Paper Award, 10th International Conference on Algorithms and Architectures for Parallel Processing, 2010

Strategy-Proof Dynamic Resource Pricing of Multiple Resource Types on Federated Clouds

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. There is growing interest in large-scale resource sharing with emerging architectures such as cloud computing, where globally distributed and commoditized resources can be shared and traded. Federated clouds, a topic of recent interest, aims to integrate different types of cloud resources from different providers, to increase scalability and reliability. In federated clouds, users are rational and maximize their own interest when consuming and contributing shared resources, while globally distributed resource supply and demand changes as users join and leave the cloud dynamically over time. In this paper, we propose a dynamic pricing scheme for multiple types of shared resources in federated clouds and evaluate its performance. Fixed pricing, currently used by cloud providers, does not reflect the dynamic resource price due to the changes in supply and demand. Using simulations, we compare the economic and computational efficiencies of our proposed dynamic pricing scheme with fixed pricing. We show that the user utility is increased, while the percentage of successful buyer requests and the percentage of allocated seller resources is higher with dynamic pricing.

1 Introduction

Currently, several technologies such as grid computing and cloud computing among others, are converging towards federated sharing of computing resources [11]. In these distributed systems, resources are commodities and users can both consume and contribute with shared resources. In cloud computing [12], resources are provided over the Internet on-demand, as a service, without the user having knowledge of the underlying infrastructure. *Public clouds* are available to all users, while *private clouds* use similar infrastructure to provide services for users within an organization. At the present, several companies such as Amazon [1], Rackspace Cloud [5], and Nirvanix [3], provide computing and storage services, using pay-per-use fixed pricing, and new capabilities, such as .NET and database services [2] are expected in the near future. Cloud computing usage is increasing both in breadth, such as the number of resource types offered, and in depth, such as the number of resource providers. Thus, with an increasing number of cloud users, it is expected that more providers will offer similar services. Furthermore, with interoperability between different providers [7], users will able to use the same service across clouds to improve scalability and reliability. In this context, the aim of *federated clouds*, a topic of recent interest, is to integrate resources from different providers such that access is transparent to the user.

A fundamental problem in any federated system is the allocation of shared resources. Recent work in distributed systems acknowledges that users sharing resources are self-interested parties with their own goals and objectives [28, 20, 16]. Usually, these parties can exercise their partial or complete autonomy to achieve their objectives and to maximize their benefit. They can devise strategies and manipulate the system to their advantage, even if by doing so they break the rules of the system. To manage rational users, economics [30] and mechanism design [24] offer market-based approaches for pricing and allocation of shared resources. Although we cannot assume rational users are trusted to follow the algorithm or protocols designed and deployed, we can assume that they participate in sharing in order to maximize their personal gain, such that incentives may be used to induce the desired behaviour. Mechanism design studies how to structure incentives such that users behave according to protocols. Thus, recent work in peer-to-peer networking [28, 15], grid or cluster computing [20], Internet routing [18], general graph algorithms [17], and resource allocation [10,31], use a form of incentives to manage rational users.

In this paper we discuss a dynamic pricing scheme suitable for allocating resources on federated clouds, where pricing is used to manage rational users. A rational user may represent either an individual user, a group, or an organization, depending on the application context. In federated clouds, users request more than one type of resources from different providers. In contrast to fixed pricing, where users have to manually aggregate resources from different providers, our pricing scheme is designed to *allocate a request for multiple resource types*. Moreover, in a federated cloud, resource demand and supply fluctuate as users join and leave the system. We show using simulations that using the proposed *dynamic* scheme, the user welfare, the percentage of successful requests, and the percentage of allocated resources is higher than using fixed pricing.

The remainder of this paper is structured as follows. Section 2 presents related works from grid computing and distributed systems. We discuss dynamic pricing for cloud computing and federated clouds in Section 3. Our auction framework is introduced in Section 4, while in Section 5 we evaluate the economic efficiency, the individual user welfare, the impact of multiple resource types and computational efficiency, measured by the computational time incurred by the proposed algorithm. Finally, Section 6 contains our conclusions and discusses our future work.

2 Related Works

Resource markets have been previously proposed for sharing computational resources in the presence of rational users [32, 30, 31, 14, 26, 19]. A *resource market* consists of the environment, rules and mechanisms where resources are exchanged. In this context, related works have used either bartering or pricing to exchange resources. In bartering, resources are exchanged directly, without using any form of currency. For example, in BOINC [9], users donate their CPU cycles by running a software client which polls a server for new jobs. In BitTorrent [15], rational users that behave selfishly and do not cooperate in sharing files are punished by other users. In contrast, in OurGrid [10], each user keeps track of other users that provide resources for their jobs, and prioritize their requests when their own resources are idle. Bartering is simple to implement and allows several types of incentives for rational users: moral incentives (volunteer computing) or coercive incentives (tit-for-tat, network of favors). However, bartering allows exchanges of a single resource type. For example, BitTorrent exchanges blocks from the same file, OurGrid is used for CPU cycles, etc. In order to exchange different types of resources, pricing and a common currency is used to express the value of each resource type.

Pricing is the process of computing the exchange value of resources relative to a common form of currency. Economic models for the allocation of shared resources may use fixed or dynamic pricing. When using fixed pricing, each resource type has a predefined price, set by the seller. For example, Amazon provides disk space for $0.15/GB. In contrast, when using dynamic pricing, the resource price is computed for each request according to the pricing mechanism used. More specifically, a resource type can have the same price for all resource providers (non-discriminated pricing), or payment is computed differently for each resource provider (discriminated pricing). Pricing schemes use financial incentives in addition to payments to motivate rational users to be truthful.

Several market-based allocation systems for grids, such as Sorma [22] and Nimrod/G [13], 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. The seller attempts to maximize the resource price, while the buyer strives to minimize it. However, communication constitutes the main disadvantage of bargaining: 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 grow further when a buyer requires more than one resource types. Thus, scalability becomes a major issue when increasing the number of users or resource types in a request.

In contrast to resource sharing systems used in research and academic communities or for personal benefit, cloud computing has been put into commercial use and its economic model is based on pricing. Previous unsuccessful cloud computing attempts, such as Intel Computing Services, required users to negotiate written contract and pricing. However, current online banking and currency transfer technologies allow cloud providers to use fixed pricing, with buyer payments made online using a credit-card. Federated clouds can be formed by combining private clouds to provide users with resizeable and elastic capacities [11]. Currently, companies such as Amazon operate as standalone clouds service providers. However, in a federated cloud, any globally distributed user can both offer and use cloud services. A user is either an individual, a group, or an organization, depending on the application context.

3 Market-Based Pricing Mechanisms

Market-based resource allocation mechanisms based on pricing introduce several economic and computational challenges. From a computational perspective, a mechanism must compute in polynomial time the allocation of multiple resource types while maximizing the number of allocated resources and satisfied requests. However, an optimal allocation mechanism for multiple resource types such as combinatorial auctions requires a NP-complete algorithm [23]. Accordingly, many systems share only one resource type, such as CPU cycles in volunteer computing, and file blocks in file-sharing.

From an economic perspective, the desirable properties for resource allocation are: *individual rationality*, *incentive compatibility*, *budget balance* and *Pareto efficiency*

[21]. In an individual rational allocation mechanism, rational participants gain higher utility by participating in resource sharing than from avoiding it. Incentive compatibility ensures that the dominant strategy for each participant is truth-telling. Budget-balance verifies that the sum of all payments made by buyers equal the total payments received by the sellers. *Pareto efficiency*, the highest economic efficiency, is achieved when, given an allocation, no improvement can be made that makes at least one participant better off, without making any other participant worse off. However, according to the Myerson-Sattherwithe impossibility theorem [21], no mechanism can achieve all four properties together. Accordingly, related works have traded incentive compatibility [14, 30], economic efficiency [19] or budget-balance [23].

Our approach is designed to achieve individual rationality, incentive compatibility and budget balance using a computationally efficient algorithm that can allocate buyer requests for multiple resource types.

In a resource market, with a large number of providers (sellers) and users (buyers), fixed pricing does not reflect the current market price resource price due to the changing demand and supply. This leads to lower user welfare and to imbalanced markets, e.g. under-demand. Figure 1 shows the welfare lost by a seller that uses fixed pricing. In the case of under-demand, the fixed price tends to be higher than the market price and buyers may look for alternative resources. In the case of over-demand, the fixed price limits the seller welfare, which could be increased by using a higher resource price.

In a federated clouds market, *dynamic pricing* sets resource payments according to the forces of demand and supply. Moreover, the use of dynamic pricing facilitates sellers to provide multiple resource types. Early cloud services such as Sun Grid Compute Utility were restricted to one resource type, e.g. CPU time [8]. More recent services, such as Amazon S3 and EC2, introduced more resource types, i.e. storage and bandwidth. Currently, Amazon has expanded its offer to 10 different virtual machine instance configurations, with different prices for each configuration, and practice tiered pricing for storage and bandwidth [1]. We see this as the first step towards dynamic pricing, where users can request for custom configurations with multiple resource types.

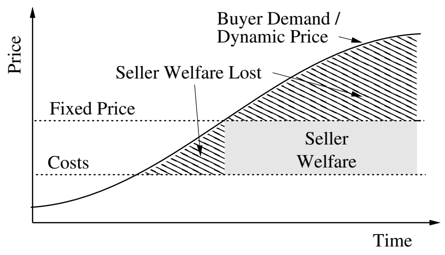

Fig. 1. Fixed Pricing Limits Seller Welfare

Resource_Type = *Description*
Publish = *Seller_Address*, *Resource_Type*, *Items*, *Cost*
Request = *Buyer_Address*, (*Resource_Type*, *Items*)+, *Price*

Fig. 2. Simplified Model for Multiple Resource Types Buyer Request

The resource market in federated clouds consists of many *resource types*. Figure 2 shows a simplified resource model where a buyer request can consist of many resource types and many resource items for each type. A resource type is loosely defined, and can be a hardware resource, a service, or a combination. We consider the example of a New York Times employee that used 100 EC2 instances to convert 4 TB of TIFF files to the PDF format [6]. To complete his job, the user required multiple resource types (storage from Amazon S3 and computational power from Amazon EC2), and multiple items (100 Amazon EC2 instances) to complete his task. In this example, we assume Amazon EC2 provides ten resource types, called *instances*. A *small instance* consists of 1 EC2 compute unit (approx. 1 GHz CPU), 1.7 GB memory and 160 GB storage, and is priced at $0.10/hour, while an *extra-large instance* consists of 8 EC2 compute units, 15 GB memory and 1.6 TB storage, and is priced at $1.00/hour.

4 Achieving Strategy-Proof Resource Pricing

In the context of federated clouds, we propose a strategy-proof dynamic pricing mechanism for allocating shared resources with multiple resource types. We assume a federated cloud resource market where rational users can both provide (sellers) and utilize resources (buyers). Rational users represent either an individual or an organization. Interoperability provides the buyers with uniformity and elasticity. Thus, a buyer request for a large number of resources can be met by more than one seller.

In a previous paper, we propose a mechanism design problem which describes a resource sharing system where rational users can be both buyers and sellers of resources [29]. Given a set of alternative choices, a rational user selects the alternative that maximizes the expected value of his utility function. In our mechanism design problem, the utility functions are determined by the seller costs and the buyer budget, respectively.

Definition (The Market-based Resource Allocation Problem). *Given a market containing requests submitted by buyers and resources offered by sellers, each participant is modeled as a rational user i with private information t_i. A seller has private information t_s^r, the underlying costs for the available resource r, such as power consumption, bandwidth costs, etc. The buyer's private information is t_b^R, the maximum price the buyer is willing to pay such that resources are allocated to satisfy its request R. Seller's i valuation is t_i^r if the resource r is allocated, and 0 if not. Similarly, buyer's i valuation is t_i^R of the request R is allocated, and 0 if not. For a particular request R, the goal is to allocate resources such that the underlying costs are minimized.*

To address the market-based resource allocation mechanism design problem in which sellers and buyers are rational participants, we propose a reverse auction-based mechanism, which we prove formally to be individual rational, incentive compatible and budget-balanced. A mechanism that is both individual rational and incentive compatible is known as *strategy-proof*.

Auctions are usually carried out by a third party, called the *market-maker*, which collects the bids, selects the winners and computes the payments. Since this paper focuses on the economical and computational advantages of dynamic pricing, we consider for

```
Allocate()
  while request = request_queue.dequeue()
    // determine winners
    DeterminWinners(request, resource_list,
                    &winners, &$c_{M|s}$)
    foreach seller in winners
      // determine $c_{M|s=\infty}$
      resource_list = resource_list − seller.resources
      DetermineWinners(request, resource_list,
                       nil, &$c_{M|s=\infty}$)
      // determine $c_{M|s=0}$
      $c_{M|s=0} = c_{M|s}$
      foreach Resource Type rt in request
        $c_{M|s=0} = c_{M|s=0}$ − seller.rt.items ∗ seller.rt.price
      end foreach
      payment = $c_{M|s=\infty} - c_{M|s=0}$
      payments.add(seller, payment)
    end foreach
    // if the request is not allocated
    // put it back in queue
    if (request.price > payments.total)
      then request_queue.enqueue(request)
    end if
  end while
end Allocate
```

```
DetermineWinners(request, resource_list,
                 &winners, &payments)
  foreach Resource Type rt in request as subreqest
    resources = filter (resource_list, Resource Type rt)
    priceSort (resources)
    while subreqest.items > 0 do
      // determine sellers
      seller = resources[rt].head().owner
      if seller.rt.items ≥ subreqest.items
        then itemsno = subrequest.items
        else itemsno = seller.rt.items
      end if
      seller.rt.items = seller.rt.items − itemsno
      subreqest.items = subrequest.items − itemsno
      winners.add(seller, itemsno)
      payments.add(seller, itemsno ∗ seller.rt.price)
    end while
  end foreach
end DetermineWinners
```

Fig. 3. Dynamic Resource Pricing Algorithm

simplicity a centralized market-maker, to which sellers publish resources, and buyers send requests. In order to improve scalability, Section 6 shows our insights into distributed auctions, where more than one market-makers are able to auction at the same time. Given that buyers and sellers are globally distributed, it is practical to adopt a peer-to-peer approach, where, after pricing and allocation, buyers connect to sellers to use the resources paid for.

The reverse auction contains two steps, *winner determination* and *payment computation*. Winner determination decides which sellers are selected for allocation, based on the published price, such that the underlying resource costs are minimized. However, to achieve strategy proof using financial incentives, the actual payments for the winning sellers are determined in the second step, based on the market supply for each resource type.

The payment for a seller is determined for each resource type using a VCG-based [17] function, which verifies the incentive-compatibility property for sellers:

$$p_s = \begin{cases} 0, & \text{if seller } s \text{ does not contribute with} \\ & \text{resources to satisfy the request} \\ c_{M|s=\infty} - c_{M|s=0} & \\ & \text{if seller } s \text{ contributes with} \\ & \text{resources to satisfy the request} \end{cases} \quad (1)$$

where:
$c_{M|s=\infty}$ is the lowest cost to satisfy the request without the resources from seller s;
$c_{M|s=0}$ is the lowest cost to satisfy the request when the cost of resources from seller s resources is 0.

To achieve the incentive-compatibility property for buyers, we select the requests using the first-come-first-serve strategy [29]. To obtain budget-balance, the buyer payment function is the sum of all seller payments:

$$p_b = -\sum_{s \in S} p_s \tag{2}$$

where S is the set of winner sellers.

Figure 3 shows the auction algorithm implemented by the centralized auctioneer. Requests are sorted in a queue according to their arrival times and are processed according to the first-come-first-serve policy (line 2). Next, the market-maker solves the winner determination problem (line 4). Payments are computed for each winner (line 6) by determining $c_{M|s=\infty}$ (line 9) and $c_{M|s=0}$ (line 11). Finally, allocation may take place if the buyer price is higher than the buyer payment, which is the sum of seller payments (line 29). Winner determination (line 26) finds the best sellers for all resource types (line 28) based on the published resource item price.

5 Impact of Dynamic Pricing

We evaluate the proposed pricing mechanism both for economic and computational efficiency. Using simulation, we compare our dynamic pricing scheme with fixed pricing, currently used by many cloud providers.

We implement our framework as an application built on top of FreePastry [27], an open-source DHT overlay network environment. FreePastry provides efficient lookup, i.e. in $O(\log N)$ steps, where N is the number of nodes in the overlay. In addition, FreePastry offers a discrete-event simulator which is able to execute applications without modification of the source code. This allows us both to simulate large systems[1], and to validate the results in a deployment over PlanetLab [4].

For simplicity, we use a centralized market-maker to compare the economic and computational advantages of dynamic pricing. A centralized implementation has the advantage of allowing the measurement of economic and computational efficiency with a simple setup for a large simulated network. Moreover, the use of a peer-to-peer substrate such as FreePastry allows us to address the scalability issue in our future work. Thus, our simulated environment contains one market-maker and 10,000 nodes, where each node can be seller and buyer. Publish and request messages are sent to the market-maker node using the FreePastry routing process, which then performs the reverse auctions using the first-come-first-serve policy and computes the payments using the algorithm in Figure 3.

5.1 Economic Efficiency

Traditionally, efficiency in computer science is measured using system-centric performance metrics such as the number of completed jobs, average system utilization, etc. All user applications are equally important and optimizations ignore the user's valuation for resources. Thus, resource allocation is unlikely to deliver the greatest value to

[1] In our experiments, we were able to use up to 35,000 peers in the FreePastry simulator.

the users, especially when having a limited amount of resources. In contrast, economic systems measure efficiency with respect to user's valuations for resources (utility). Consequently, in a Pareto efficient system, where economic efficiency is maximized, a user's utility cannot improve without decreasing the utility of another user.

Economic efficiency is a global measure and represents the *total* buyer and seller welfare. More specifically, there are two factors that affect the economic efficiency: *i)* average user welfare; and *ii)* number of successful requests, for buyers, and number of allocated resources, for sellers.

Using fixed pricing, the average user welfare is constant, since the user utility is also constant. In contrast, when using dynamic pricing, the average user welfare fluctuates with the computed payments, according to the resource demand. Moreover, a dynamic pricing scheme is able to balance the number of successful requests and the number of allocated resources depending on the market condition. For example, resource contention in the case of over-demand is balanced by increasing the resource price. Similarly, buyers are incentivized by a lower price when the market condition is under-demand. Overall, dynamic pricing achieves better economic efficiency both with higher average user welfare, and a higher number of successful buyer requests and allocated seller resources.

5.2 User Welfare

The *user welfare* is determined by the difference between the user utility and payment. In our proposed framework, the user utility is the same as the published price, since both buyers and sellers are truthful, according to the incentive compatibility property of our pricing algorithm. In the case of fixed pricing, we also consider a truthful buyer, i.e. the published request price represents the buyer's utility. However, we do not make the same assumption about sellers, which have a fixed resource price that may differ from the seller's utility. Thus, in our experiment, we compare only the average buyer welfare when using fixed and dynamic pricing, respectively.

For this experiment, we consider a balanced market, where supply and demand are equal. Thus, we assume that the market-maker receives events with an interarrival time of $1s$, where an event has equal probability of being a buyer request or a seller resource publish. Events are uniformly distributed between 10,000 FreePastry nodes, and contain of a number of resource types uniformly distributed between 1 and 3, chosen randomly from a total of 5 resource types. The number of items for each resource type is generated

Table 1. Dynamic Pricing Increases Buyer Welfare

Metric	**%Price Variation**	**Pricing Scheme**	
		Fixed	*Dynamic*
avg buyer welfare	10	3.5	4.6
	20	7.4	9.3
	50	18.8	23.3
%succ buyer	10	47.7	62.5
	20	48.8	62.2
	50	49.5	62.1

according to an exponential distribution with mean 10. For sellers, we assume 100 as the fixed price, while in the case of dynamic pricing we vary the price by 10%, 20% and 50%, i.e. the price is generated according to a uniform distribution between 90 and 110, 80 and 120, and 50 and 150, respectively. Buyer price is varied according to the same percentage, shown in Table 1 as *%Price Variation*. The simulation runs for 600,000 events, which, for an arrival rate of $1s$, give a total simulation time of approximately seven days. To reduce sampling error, we run our experiments three times and compute the average.

The results in Table 1 show that dynamic pricing increases the buyer welfare and the percentage of successful buyer requests (*%succ buyer*). Given that the mean buyer utility is 100, and a *theoretical*(the actual maximum welfare can be computed using a NP-complete algorithm, and is smaller than the *theoretical* welfare.) maximum welfare for an item is achieved when having the minimum payment, i.e. $100-$*Price Variation*, we derive that the maximum welfare equals the price variation. Thus, using the proposed dynamic pricing mechanism increases buyer welfare by approximately 10%, when compared to fixed pricing.

5.3 Multiple Resource Types in Different Market Conditions

In contrast to fixed pricing, where users have to manually aggregate resources, the proposed dynamic pricing scheme can allocate buyer requests for multiple resource types. However, with the increase in the number of resource types in the request, it is reasonable to assume that the overall user welfare will decrease. Another factor that influences the user welfare is the market condition. Thus, when there is under-demand, the buyer welfare should increase. Similarly, the seller welfare should increase when there is over-demand. Next, we study the influence of multiple resource types and different market conditions for the proposed dynamic scheme and compare to fixed pricing.

We vary the number of resource types in a request to 5, 10, and 20, while the price variation is set to 20%. We consider 3 market conditions: *Under-Demand*, when supply

Table 2. Dynamic Pricing Increases Efficiency For Multiple Resource Types

Resource Types	**%succ buyer**		**%succ seller**		**avg seller welfare**		**avg buyer welfare**	
	fixed	*dynamic*	*fixed*	*dynamic*	*fixed*	*dynamic*	*fixed*	*dynamic*
Under-Demand								
5	48.4	82.3	24.1	41.8	N/A	2.9	6.2	10.9
10	47.4	86.3	23.4	44.1	N/A	3.0	4.7	9.4
20	46.5	89.7	22.1	46.4	N/A	3.2	3.3	8.1
Balanced Market								
5	48.2	62.4	47.5	61.0	N/A	4.5	6.2	7.9
10	47.1	62.9	46.5	62.5	N/A	4.6	4.7	6.3
20	46.2	63.3	46.0	64.0	N/A	4.7	3.4	4.9
Over-Demand								
5	48.2	42.1	95.4	75.5	N/A	5.9	6.2	6.1
10	47.4	41.4	93.1	74.2	N/A	5.8	4.7	4.8
20	46.2	40.4	91.7	73.0	N/A	5.6	3.4	3.7

is greater than demand, *Balanced Market*, when supply equals demand, and *Over-Demand*, when supply is less than demand. To simulate different market conditions, we vary the probability of a request event. Thus, in the case of a balanced market, the probability is set to 50%, while for under-demand is 33%, and for over-demand is 66%. We measure economic efficiency, i.e. *avg seller welfare* and *avg buyer welfare*, and pricing scheme performance, i.e. *%succ buyer* and *%succ seller*. Table 2 presents our results.

In the case of fixed pricing, the percentage of successful buyer request is close to 50% for all market conditions, since the buyer item price is uniformly distributed with the mean equal to the seller item price. However, the percentage of successful buyer requests decreases when the number of resource types increases since the number of sellers that are allocated to satisfy a request also increases.

In contrast, when using dynamic pricing, the percentage of successful buyer requests varies under different market conditions, according to the forces of supply and demand. Thus, when supply is greater than demand, the percentage of successful buyer requests is higher than in the case of a balanced market, while for over-demand the percentage decreases further. Using the proposed auction mechanism achieves a higher percentage of successful buyer requests and seller allocated resources, in the case of under-demand and balanced market. When demand is higher than supply and the number of resource types in a request increases, there is premise for monopolistic sellers [25], i.e. there are not enough sellers in the market to compute payments using a VCG-based payment function such as the seller payment used by our auction framework.

Similarly, using fixed pricing results in the same mean buyer welfare when varying market conditions, and welfare decreases when increasing the number of resource types. For the proposed pricing mechanism, the buyer welfare is higher when compared to fixed pricing, and varies according to supply and demand: buyer welfare increases when supply is greater than demand, and decreases when demand is higher.

5.4 Cost of Dynamic Pricing

Computational efficiency is a major design criteria in the allocation of shared resources. Optimal mechanisms such as combinatorial auctions [23] are not feasible since the winner determination algorithm is NP-complete. Fixed pricing has the advantage of eliminating the payment computation. To determine the time cost of the algorithm used by the proposed mechanism, we analyze the run-time complexity of the winner determination and the buyer and seller payment functions. Without considering queuing time, the total time incurred by our mechanism is then:

$$T = T_{wd} + T_p$$

where T_{wd} is the time taken to determine the winners, and T_p is the time for the payment computation. We consider the following inputs: RT, the number of resource types in a request; I_{RT_k}, the number of items from the resource type RT_k in a request; S_{RT_k}, the number of sellers with resource type RT_k. We use $\sum_k S_{RT_k}$ to represent the total number of published resources.

The winner determination algorithm in Figure 3 contains 2 loops (lines 28 and 31) for the number of resource types in the request, and the number of items of each resource type, while the inner code (lines 32–41) takes a constant amount of time. Finding all resources with the same type (line 29) depends on $\sum_k S_{RT_k}$, while sorting resources according to their price takes $O(S_{RT_k} \log S_{RT_k})$. Thus, in the worst case, the complexity of the winner determination algorithm is:

$$T_{wd} = O(RT \times (\sum_k S_{RT_k} + S_{RT_k} \log S_{RT_k} + I_{RT_k})) \tag{3}$$

Similarly, we compute the complexity of the payment algorithm (lines 6–18), which, in the worst case scenario, is: $T_p = O(I_{RT_k} \times T_{wd})$, when each winner seller provides one item. Thus, the total time taken by the proposed allocation algorithm is:

$$\begin{aligned} T = T_{wd} + O(I_{RT_k} \times T_{wd}) = O(I_{RT_k} \times T_{wd}) = \\ = O(I_{RT_k} \times RT \times (\sum_k S_{RT_k} + S_{RT_k} \log S_{RT_k} + I_{RT_k})) \end{aligned} \tag{4}$$

In conclusion, the complexity of the algorithm used by the proposed framework is a polynomial function of the number of resource types in a request, RT; the number of items requested for each resource type, I_{RT_k}; the total number of published resources, $\sum_k S_{RT_k}$; and the number of sellers with resource type k, S_{RT_k}.

Figure 4 shows the mean request allocation time obtained in our simulations. We run the simulator on a quad-core Intel Xeon 1.83 GHz CPU with 4 GB of RAM. The figure shows the increase in allocation time due to the increase in number of resource types, and the different market scenarios. In the case of over-demand, the number of sellers is smaller and, consequently, the allocation time is smaller. Similarly, in the case of under-demand, the allocation time increases.

While the allocation time for a small number of resource types is under 1 second, a large number of resource types in the system leads to increased allocation times. Thus,

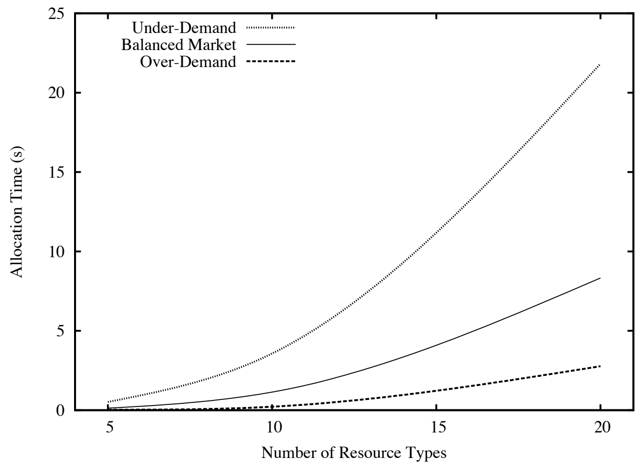

Fig. 4. Allocation Time When Varying Resource Types Under Different Market Conditions

scalability becomes an issue both when the number of resource types, and the number of sellers increases. To improve vertical scalability, i.e. when the number of resource types increases, we are currently developing a distributed framework, where a peer-to-peer network of nodes perform reverse auctions for different resource types at the same time. Furthermore, to address horizontal scalability, i.e. when the number of users increases, different peers maintain separate resource lists and request queues, such that allocation of requests for different resource types is parallelized.

6 Conclusions and Future Work

In this paper, we discuss current resource allocation models for cloud computing and federated clouds. We argue that dynamic pricing is more suitable for federated sharing of computing resources, where rational users may both provide and use resources. To this extent, we present an auction framework that uses dynamic pricing to allocate shared resources. We define a model in which rational users, classified as buyers and sellers, trade resources in a resource market. Our previous paper shows that the payment mechanism used by the proposed auction framework is individual rational, incentive compatible and budget balanced. In this paper we use the defined model to study both the economic and computational efficiency of dynamic pricing, in the context of federated clouds. Our focus is a dynamic pricing scheme where a buyer request consists of multiple resource types. We implement our framework in FreePastry, a peer-to-peer overlay, and use the FreePastry simulator to compare our dynamic pricing mechanism with fixed pricing, currently used by many cloud providers. We show that dynamic pricing increases the buyer welfare, a measure of economic efficiency, while performing a higher number of allocations, measured by the percentage of successful buyer requests and allocated seller resources.

Even though the auction algorithm is polynomial, scalability becomes an issue as the number of resource types in a request increases. We are currently implementing a scheme that uses distributed auctions, where multiple auctioneers can allocate different resource types at the same time. Specifically, by taking advantage of distributed hash tables, we aim to create an overlay peer-to-peer network which supports resource discovery and allocation using the proposed dynamic pricing mechanism.

Acknowledgments

This work is supported by the National University of Singapore under grant number R-252-000-339-112.

References

1. Amazon Web Services (2009), `http://aws.amazon.com`
2. Microsoft Azure Services Platform (2009), `http://www.microsoft.com/azure`
3. Nirvanix Storage Delivery Network (2009), `http://nirvanix.com`
4. An open platform for developing planetary-scale services (2009), `http://planetlab.org`

5. The Rackspace Cloud (2009), http://www.rackspacecloud.com
6. Self-service, prorated super computing fun (2009), http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun
7. Sun Cloud Computing Initiative (2009), http://www.sun.com/solutions/cloudcomputing
8. Sun Grid Compute Utility (2008), http://www.network.com
9. Anderson, D.P.: BOINC: A System for Public-Resource Computing and Storage. In: 5th IEEE/ACM Intl. Workshop on Grid Computing, Pittsburgh, USA, pp. 4–10 (2004)
10. Andrade, N., Cirne, W., Brasileiro, F.V., Roisenberg, P.: OurGrid: An Approach to Easily Assemble Grids with Equitable Resource Sharing. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2003. LNCS, vol. 2862, pp. 61–86. Springer, Heidelberg (2003)
11. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, USA (2009)
12. Buyya, R., Yeo, C.S., Venugopal, S.: Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In: Proc. of the 10th IEEE Intl. Conf. on High Performance Computing and Communications, Dalian, China, pp. 5–13 (2008)
13. Buyya, R., Abramson, D., Giddy, J.: Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid. In: Proc. of the 4th Intl. Conference on High Performance Computing in Asia-Pacific Region, Beijing, China, pp. 283–289 (2000)
14. Chun, B.N., Culler, D.E.: Market-based Proportional Resource Sharing for Clusters. Technical Report UCB/CSD-00-1092, EECS Department, University of California, Berkeley, USA (2000)
15. Cohen, B.: Incentives Build Robustness in BitTorrent. In: Proc. of the 1st Workshop on Economics of Peer-to-Peer Systems, Berkeley, USA (2003)
16. Dani, A.R., Pujari, A.K., Gulati, V.P.: Strategy Proof Electronic Markets. In: Proc. of the 9th Intl. Conference on Electronic Commerce, Minneapolis, USA, pp. 45–54 (2007)
17. Elkind, E.: True Costs of Cheap Labor Are Hard to Measure: Edge Deletion and VCG Payments in Graphs. In: Proc. of the 7th ACM Conference on Electronic Commerce, Vancouver, Canada, pp. 108–116 (2005)
18. Feigenbaum, J., Papadimitriou, C.H., Shenker, S.: Sharing the Cost of Multicast Transmissions. Journal of Computer and System Sciences 63, 21–41 (2001)
19. Lai, K., Huberman, B.A., Fine, L.R.: Tycoon: A Distributed Market-based Resource Allocation System. Technical Report cs.DC/0404013, HP Labs, Palo Alto, USA (2004)
20. Lin, L., Zhang, Y., Huai, J.: Sustaining Incentive in Grid Resource Allocation: A Reinforcement Learning Approach. In: Proc. of the IEEE Intl. Symposium on Cluster Computing and the Grid, Rio de Janeiro, Brazil, pp. 145–154 (2007)
21. Myerson, R.B., Satterthwaite, M.A.: Efficient Mechanisms for Bilateral Trading. Journal of Economic Theory 29(2), 265–281 (1983)
22. Nimis, J., Anandasivam, A., Borissov, N., Smith, G., Neumann, D., Wirstrm, N., Rosenberg, E., Villa, M.: SORMA - Business Cases for an Open Grid Market: Concept and Implementation. In: Altmann, J., Neumann, D., Fahringer, T. (eds.) GECON 2008. LNCS, vol. 5206, pp. 173–184. Springer, Heidelberg (2008)
23. Nisan, N.: Bidding and Allocation in Combinatorial Auctions. In: Proc. of the 2nd ACM Conference on Electronic Commerce, Minneapolis, USA, pp. 1–12 (2000)
24. Nisan, N., Ronen, A.: Algorithmic Mechanism Design (extended abstract). In: Proc. of the 31st Annual ACM Symposium on Theory of Computing, Atlanta, USA, pp. 129–140 (1999)

25. Pham, H.N., Teo, Y.M., Thoai, N., Nguyen, T.A.: An Approach to Vickrey-based Resource Allocation in the Presence of Monopolistic Sellers. In: Proc. of the 7th Australasian Symposium on Grid Computing and e-Research (AusGrid 2009), Wellington, New Zealand, pp. 77–83 (2009)
26. Regev, O., Nisan, N.: The Popcorn Market: An Online Market for Computational Resources. In: Proc. of the 1st Intl. Conference on Information and Computation Economies, Charleston, USA, pp. 148–157 (1998)
27. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object address, and Routing for Large-Scale Peer-to-Peer Systems. In: Proc. of the IFIP/ACM Intl. Conference on Distributed Systems Platforms, Heidelberg, Germany, pp. 329–350 (2001)
28. Shneidman, J., Parkes, D.C.: Rationality and Self-Interest in Peer to Peer Networks. In: Kaashoek, M.F., Stoica, I. (eds.) IPTPS 2003. LNCS, vol. 2735, pp. 139–148. Springer, Heidelberg (2003)
29. Teo, Y.M., Mihailescu, M.: A Strategy-proof Pricing Scheme for Multiple Resource Type Allocations. In: Proc. of the 38th Intl. Conference on Parallel Processing, Vienna, Austria, pp. 172–179 (2009)
30. Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: Analyzing Market-Based Resource Allocation Strategies for the Computational Grid. International Journal of High Performance Computing Applications 15(3), 258–281 (2001)
31. Wolski, R., Plank, J.S., Brevik, J., Bryan, T.: G-commerce: Market Formulations Controlling Resource Allocation on the Computational Grid. In: Proc. of the 15th Intl. Parallel and Distributed Processing Symposium, San Francisco, USA, pp. 46–54 (2001)
32. Yeo, C.S., Buyya, R.: A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing. Software: Practice and Experience 36, 1381–1419 (2006)