Resource Allocation in Federated Distributed Computing Infrastructures

Alvin AuYoung[†,*], Brent N. Chun[‡], Alex C. Snoeren[†], and Amin Vahdat[†]

[†] UC San Diego [‡] Intel Research Berkeley

1 Introduction

We consider the problem of allocating combinations of heterogeneous, distributed resources among self-interested parties. In particular, we consider this problem in the context of distributed computing infrastructures, where resources are shared among users from different administrative domains. Examples of such infrastructures include PlanetLab [15] and computational grids [7].

End-users derive utility from receiving a share of resources. When there is an excess demand for resources, it isn't possible to completely satisfy all resource requests. Therefore, we argue that it is important for these infrastructures to allocate resources in a way that maximizes aggregate end-user utility. Such an allocation system is known as *economically efficient*. Because a user's utility function for resources isn't typically known a priori, determining an allocation policy to maximize utility is difficult in the presence of excess demand. As use of these infrastructures becomes more widespread, contention for resources will increase, and allocating resources in an economically efficient manner becomes more difficult. Figure 1 shows a snapshot of resource demand in PlanetLab. Due to the way resources are distributed in PlanetLab, the rise in contention decreases the portion of resources received by any individual user, thereby reducing the amount of useful work that can be completed in the system.

We argue that given the appropriate mechanisms, end-users can cooperate to arrive at an optimal resource allocation in spite of excess demand. More specifically, by allowing users to express preferences for when and what resources are received, the system will be able to spread out some excess demand over time, and increase the efficiency of the system. To this end, we present the design and early implementation of Bellagio, a distributed resource discovery and market-based allocation system for federated distributed computing infrastructures. In this system, users identify resources of interest using a resource discovery mechanism, and express preference for these resources over time and space in the form of combinatorial auction bids. Resource allocation is controlled by a centralized auctioneer. In addition, Bellagio uses a strategy-proof design [5] that provides incentive for end-users to reveal their true valuation of resources.

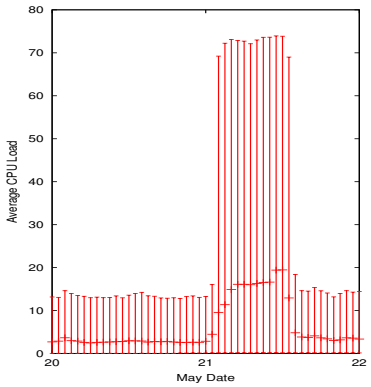

Figure 1: 5th, average and 95th percentile load average on 220 PlanetLab nodes leading up to the May 2004 OSDI deadline.

To our knowledge, this is the first system that supports allocation of combinations of heterogeneous goods in a flexible and economically efficient manner. We have implemented a prototype and validated its performance through simulation. We plan to deploy a complete system on the PlanetLab wide-area test bed to gain experience with real users.

2 Related work

Proportional-share scheduling is a common approach to provide access to local, time-shared resources. This is an effective approach, but is insufficient if

*This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.

employed as the only mechanism for resource sharing. For example, if users were to behave in a conscientious manner, they would use their share of the resources only when needed. This behavior allows resources to be used more productively. However, if users are self-interested, they will always consume their share of available resources, thereby lowering the amount of useful work performed by the system.

Market-based approaches have been applied to motivate efficient use of resources. Users have a finite amount of virtual currency and have an incentive to consume resources sparingly [19, 22]. The primary limitation in this model is the difficulty in scheduling a combination of resources simultaneously, which is often desired. Co-scheduling techniques [2] have been applied to address this problem in clusters, but these approaches are ineffective when resources are differentiated and spread over a wide area. More importantly, this technique does not allow the end-user to express an interest in a particular combination of goods.

Other market-based approaches such as Spawn [21], Popcorn [16], Tycoon [10] and SuperWeb [1], provide empirical evidence for the effectiveness of market-based mechanisms in allocating resources in an economically efficient manner. In these systems, resources are allocated in proportion to the amount of currency spent. As resources become more expensive (i.e. under heavier demand), there is incentive for users to reduce consumption. However, all of these systems rely on the scheduling of a single good: CPU cycles. We are interested in an environment where heterogeneous goods exist (i.e., disk space, memory, bandwidth) and users can express interest in particular sets of resources.

Fixed-price schemes [20, 23] deal with heterogeneous goods by calculating a market-clearing price for each resource based on historical data of the demand. (The goal of these schemes is to clear the market, which is different from maximizing aggregate end-user utility). We believe that providing the end-users the most benefit should be the priority of the system rather than maximizing resource utilization.

Market-based approaches in computational grid environments such as Nimrod/G [4] allow users to specify the types of resources needed and negotiate with the system for the use of a particular set of resources at a particular price. This requires the system to perform complex tasks such as price negotiation and resource discovery for a potentially large number of resources. This is particularly difficult if every user wishes to obtain a combination of resources.

Other approaches often used in computational grids like GRAM [9] and Condor [11] have no notion of economic efficiency when allocating resources, and provide no incentive for users to consume resources sparingly.

3 Deployment

Bellagio is intended to serve as a resource discovery and resource allocation system for distributed computing infrastructures. The first platform on which we intend to deploy Bellagio is PlanetLab, an overlay network of 440 machines hosted by 195 sites across the world. The sites are mainly research institutions from over 25 countries.

The purpose of PlanetLab is to encourage development of new, disruptive technologies to the Internet. It serves as a wide-area testbed for academic researchers, and eventually will be a deployment platform for distributed services. The value of PlanetLab as a testbed derives mainly from the global distribution of machines; academic researchers are able to test new network services under real-world network conditions. As a deployment platform, it can allow individual sites to use global resources to support a distributed client base.

Applications run on a *slice* of PlanetLab resources. A slice is a share of resources on a set of PlanetLab machines chosen by the application. Currently, each PlanetLab machine runs a Linux-based operating system and supports mechanisms that allow centralized administration of the individual machines. Administration is able to monitor the health and status of each machine, manage user accounts and distribute cryptographic keys on all PlanetLab machines.

PlanetLab machines support the notion of proportional share scheduling, where each slice is guaranteed $\frac{1}{nth}$ of a machine's resources. Admission control is performed to limit shares of a machine to no less than $\frac{1}{1000}$ (i.e. no more than 1000 simultaneous slices can run on a machine). Otherwise, there is no constraint on what resources an application's slice can bind to.

4 Bellagio architecture

Our environment can be modeled as a virtual marketplace where users spend virtual currency in exchange for combinations of system resources. We expect to be able to extend this model to an economic exchange where users can act as both producers and consumers of resources to exchange any good (resources or virtual currency) for other goods. In the current architecture, users receive a virtual currency budgets based on their resource contribution to the system.

Many of the challenges in designing a resource allocation system are mirrored in economics as economics researchers try to create mechanisms to enforce particular long-term market characteristics. This motivates the use of a market-based design for Bellagio.

Economic markets can be broadly categorized into two types: commodities markets and auctions. An auction is a much more appealing model for a market when the seller of a resource does not know the buyers' valuation for it. This is especially important because we expect demand to fluctuate rapidly at times. Requiring end-users to submit bids alleviates the system from having to determine the value (price) of combinations of goods, which can be an expensive computational task.

As a result of adopting a combinatorial auction model for our marketplace, we have designed Bellagio around the following components: resource discovery and resource market.

4.1 Resource discovery

Distributed resource discovery allows end users to identify available resources. This is particularly important when resources are heterogeneous. Bellagio uses SWORD [13] to perform resource discovery. SWORD is a distributed resource discovery service that is currently deployed on PlanetLab to allow users to locate particular resources matching various criteria. For example, users can search for resources based on resource-specific attributes (e.g., machines with low CPU load and large amounts of free memory), inter-resource attributes (e.g., machines with low inter-node latency), and logical (e.g., machines within a specific administrative domain) or physical attributes (e.g., geographic location). Along with each of these attributes, SWORD allows users to specify a utility function, which specifies the extent to which the user allows the attributes to deviate from the specified values.

SWORD returns a set of resources and how closely they match the user's specified utility functions. The end-user can perform multiple queries to determine different sets of resources and her valuation for the different sets. Additional queries can be performed to establish interdependent attributes between the resources.

Because resources are dynamic, and can exhibit changing temporal characteristics, it might not be useful for users to bid on specific resources using the results of the resource discovery mechanism. Instead, the intended use of the resource discovery mechanism is to express abstract resource specifications. Bellagio uses SWORD to translate these abstract resource specifications into concrete candidate resource sets. Users express preferences for resource specifications using the bidding language.

4.2 Resource market

Users express preferences for resources using a bidding language, and resources are allocated by running a period auction.

Bids in Bellagio are expressed in units of virtual currency. Each participating site has a bank account which maintains a balance of virtual currency available to authorized users of the participating site. As mentioned, we assume the existence of an authentication infrastructure that can authenticate bids, transfer resource capabilities and verify account balances. Bids from a participating site are only accepted as long as the balance in the associated site's account is large enough to pay the bid price.

The design of the bidding languages determines the possible bids that can be made and how easily they can be expressed by the user. The trade-off between different bidding languages is the expressiveness of the language (how many types of bids can be expressed) versus the space complexity in representing these bids. Previous work in combinatorial auction bidding languages [3, 12, 18] has examined this problem and found certain bidding languages to be more expressive and space-efficient than others.

The difficulty in designing a bidding language is the tradeoff between the expressiveness the bidding language, and the amount of work that must be performed to satisfy complex bids. In our initial implementation, we use the XOR [12]language to define our bidding language, as it seems like an intuitive language that can be understood easily by end-users, and we do not expect the cases requiring a large amount of space for bid expression to be common. We evaluate the efficacy of resolving bids in this language in our evaluation of the system.

A bid consists of sets of resources to bid for, a time period for which they are desired, and the amount of virtual currency to associate with the bid. Auctions are cleared every hour. Resources can be bound for no longer than 64 hours, and no longer than a week in advance.

Formally, bids are constructed as follows:

$$b = (s_0, s_1, d, \{q_1, q_2, ...\}, \{r_1, r_2, ...\}, v)$$

where we define the variables as follows:

A set of auctions for which to consider this bid $T = \{1, 2, 3..., 168\}$, a duration for the resource reservation, $D = \{1, 2, 3, ..64\}$, and a set of resources $N = \{r_1, ...\}$. A bid is defined as $b_1 \oplus b_2 \oplus .....b_i$,

where each b_i is a bid as previously constructed. $s_0, s_1 \in T$ defines a range of possible start times, $d \in D$, $\{r_1, r_2, ...\}$ is a resource set, constructed by the resource discovery mechanism, where $r_i \in N$, and $q_i \geq 1$ is the minimum total number of resource r_i required. v is the value of the bid as defined by the virtual currency in the system.

As mentioned earlier, the resource discovery mechanism may be used to find a specific set of resources, or it may be used to formulate a set of abstract resources by specifying a set of resource attributes. Before bids are resolved, Bellagio uses the resource discovery mechanism to populate the abstract resources in the bid with appropriate resource sets. This is especially important when resources are highly dynamic. The resource allocation would thus be performed with the most current set of resource attributes.

Our resource market allows the allocation of complex combinations of goods. Because users are self-interested, they may still be motivated to acquire more resources than they need. While providing a limited budget of virtual currency might limit this behavior, we provide general strategy-proof mechanisms [5] to provide desired behavior in our system. Strategy-proof mechanisms are based on the economic theory of mechanism design, which provides users incentive to behave in a particular way. For our purposes, this behavior is to reveal their true value for a resource, thereby allowing the system to maximize economic efficiency by allocating resources to the users who actually value them the most, and for the appropriate price. We describe the mechanisms used to allocate resources and provide strategy-proofness to Bellagio.

We employ SHARE [6] for resource allocation, mainly because it supports the exchange of heterogeneous resources. SHARE allocates resources by clearing a combinatorial auction. The winner-determination problem in a combinatorial auction is known to be NP-complete [18]. However, there exist approximation algorithms for winner determination in a combinatorial auction [17, 24]. We are investigating this use of different approximation algorithms to clear these auctions. The experimental algorithms are described in detail by Chun et al.

SHARE uses the *threshold rule* [14] to determine payments. The threshold rule is similar to the payment rule used in second-price auctions. In these auctions, the payment for a winning bid of a good is set at the price of the second-highest bid for that good. The key result of this payment rule is that it forces users to reveal their true value for goods when bidding. Since it is a bidder's dominant strategy to bid her exact valuation for a good, rational bidders will always bid truthfully, allowing SHARE to allocate resources in a way that maximizes economic efficiency. See [14] for more details.

Once the payment amount for each winning bid has been determined by the threshold rule, the user's bank account is charged by the appropriate amount, and resource bindings are performed by returning resource capabilities in the form of tickets to the winning bidders using a system such as SHARP [8]. Access to all resources in Bellagio is guarded by resource capabilities. A resource capability represents the right for an application to use a resource, and an application must present the resource capability to the resource in order to use it.

5 Implementation

This section describes our initial implementation of Bellagio. In addition to describing the user interface and algorithms for auction clearing, we discuss a set of open issues in defining appropriate policies for our system, and describe our initial approach in dealing with them.

5.1 User interface

Bellagio provides a Web-based interface as well as a command-line interface to perform resource discovery queries and construct bids. The Web-interface runs from a cluster of Linux machines and uses PHP as a front-end, and a PostgresSQL database as a back-end. The database contains account balances for virtual currency and manages user authentication. The command-line interface is provided to support the use of automated agents to perform bid placement and currency management. The interface uses XML-RPC to communicate with the centralized Bellagio server. In a live deployment, the authentication infrastructure will have to be integrated with that of the distributed computing infrastructure.

Each user of Bellagio is given an account. In the account, a user can view the results or previous resource discovery queries, bids, and upcoming auction-clearing periods.

Specifications for the resource sets used in previous resource discovery queries can be stored in the user account. In addition, the Web interface for Bellagio provides a hook into the SWORD Web interface to allow direct resource discovery queries. Resource attributes provided by SWORD are available in Bellagio to create resource specifications.

Once resource specifications are created, they can be used to construct bids. Like resource specifica-

tions, bids can be stored for later use. To place a bid, the user specifies a resource specification (or perhaps a set of resource specifications), acceptable start times, duration and bid amount. Once a bid is placed, it cannot be withdrawn once the auction-clearing period begins.

Multiple bids can be placed, but bids that overwithdraw an account are not considered. For example, if a user submits two bids, and the amount of each bid is for the complete account balance, one bid will be ignored once the other is satisfied. Bids that are invalid are discarded by the auction-clearing algorithm.

To illustrate, we present an example of an experiment wishing to analyze the effectiveness of a defense against distributed denial-of-service (DDoS) attack.

This user is interested in acquiring fractional shares on a large number of machines that are scattered across the country to simulate DDoS attackers. She would like to obtain 100 such *attacker* machines, but would be satisfied with at least 50. She is also interested in a set of 20 high-bandwidth machines to serve as proxies and 20 to server as Web servers (victims). She would also be satisfied with 10 of each.

First, to define a resource specification for attackers, she uses SWORD provides an attribute for a machine's geographic network coordinates. She provides 100 distinct geographic network coordinates as an attribute, thereby requesting a machine that is within some distance of each coordinate. Furthermore, she can also require the constraint that no more than two nodes reside within the same PlanetLab site. Using the bandwidth attribute of SWORD, she can similarly define a resource specification for the proxies and the Web servers.

The experiment is to run over the course of two days, starting any time in the next two weeks. And because she is OK with either 50 or 100 nodes, she uses the bidding language to express her valuation for each. In this example, she wishes to bid her entire virtual currency balance on the desired resource set, but would also be willing to pay half of that amount for half of the resources. This is illustrated in Table 1.

At the end of an auction, users are notified with the outcome of their bid. If a bid is satisfied, the appropriate resource capabilities are available for download from the user account in the form of an XML document [8]. In addition, an e-mail notice is sent to the user (an e-mail address is required when registering for Bellagio, notifying them of the outcome of the bid(s).

5.2 Resource market

The initial implementation of the resource market use a greedy approximation algorithms to clear the combinatorial auction. The approximation algorithm reduces the amount of time needed to compute a set of winners at the potential cost of a less efficient allocation. It calculates the value of a bid by diving the bid amount by the product of duration and number of resources, and performs allocations in decreasing order of bid values. Ties are broken by giving preference to resource sets that are more specific. The algorithm scales well in practice, requiring no more than a few minutes to resolve any auction. There is work on providing another approximation algorithm, formulated as a solver for a mixed-integer programming problem (MIP), requires more time, but can arrive at solutions that are closer to optimal for particular sets of bids. Currently this algorithm does not scale well across an increasing number of bids and resources, so we defer development of this algorithm to future work.

Bellagio clears auctions once per hour, with the possibility to reserve resources for almost three days. This is likely an important parameter to control upon deployment as it represents a tradeoff between usability (how long experiments can hope to run) and how far in advance users can plan their experiments.

5.3 Policies

One benefit of the auction model used by Bellagio is that we decouple the economic policy from the mechanisms used to enforce it. Resource capabilities are the means to obtain resource shares, and virtual currency is the means to obtain resource capabilities. By controlling the distribution of wealth in the virtual economy, Bellagio can control the share of resources received by individual participants over arbitrarily long time-scales.

In order to control the distribution of wealth among users, there are two policies that Bellagio must determine: how to earn virtual currency, and how to limit wealth.

In order to obtain resources, users must obtain virtual currency. One possibility is to back virtual currency by real currency. An alternative is to allow users to earn virtual currency by contributing resources to the shared resource pool. This has the advantage of also providing an incentive to contribute resources to the infrastructure. However, there are several potential limitations with this method. First, it is difficult to measure the contribution of a resource relative to another. For example, donating four machines with 3-GHz, hyper-threading processors might be more valuable than donating four machines with

s_0	s_1	d	$\{q_1, q_2, ...\}$	$\{r_1, r_2, ...\}$	v (virtual currency)
0	168	48	$\{100, 20, 20\}$	$\{attackers, proxies, servers\}$	all
0	168	48	$\{50, 10, 10\}$	$\{attackers, proxies, servers\}$	$all \cdot \frac{1}{2}$

Table 1: Example bid for the DDoS defense example. Each row represents a clause in an XOR bid. *all* represents the entire balance of virtual currency.

dual 33-MHz processors. One method for valuation of a contributed resource might be to measure the amount of demand for a particular resource. Unfortunately this opens up many opportunities for potential collusion since a user would then be able to artificially inflate the value of his contribution by consuming it when it would have otherwise been idle or by convincing a friendly party to consume those resources in preference to otherwise equivalent resources.

To relieve the burden of having to divide revenue earned from resources to the parties whose resources where used, we have initially adopted the model that Bellagio owns all shared resources and divides up revenue according to some predefined allocation policy. A user's contribution to the federated system can be measured out-of-band. The downside of this is that it requires an agreed-upon currency distribution policy by participating users, but we assume a fair policy can eventually be discovered through practice. The virtual nature of the currency allows sufficient latitude in discovering an agreeable distribution policy.

Assuming the existence of a currency distribution policy, we can assume that all participating sites receive periodic deposits of virtual currency in their bank accounts. This can become dangerous if a few parties decide to save currency for long periods of time in order to amass a disproportionate amount of wealth. Any amount of strategy-proof design cannot prevent such a user from dominating control over all resources since he can over bid with little consequence.

To address this issue, we bound the amount of virtual currency that can be accumulated in a bank account. The potential side effect of both approaches is a bound that is too high or too low. A cap that is too high will allow parties to amass particular amounts of wealth, and a cap that is too low will disproportionately punish the wealthier users, thereby mitigating the effects of differentiated currency distribution.

As with real-world economic policy, the level of inflation must be constantly monitored. By observing long-term resource distribution patterns among the users, the system can adapt the currency cap appropriately. While we hope that long-term distribution of resources among users in Bellagio is stable, the effects of different currency policies is likely to be unclear. We have described two policies which Bellagio can control in order to control the distribution of wealth should the observed behavior deviate from the desired behavior.

6 Evaluation

The main performance requirements for Bellagio are that it be efficient, fair, and usable.

We measure efficiency as how well client resource requests are mapped into resource bindings, and how quickly these mappings can be assigned. This metric can be thought of as how well aggregate user utility is maximized, where utility is defined as the value a resource binding has to a particular user. In economic terms, this is known as maximizing social welfare. The system also needs to be able to compute a resource allocation reasonably fast in order to be useful.

Fairness describes the variation in individual user utility provided by the system. The system should allocate resources such that the long-term utility of any individual user follows a pre-defined value relative to the utility of other individual users in the system. Furthermore, this long-term utility should not be adversely affected by the behavior of other participants in the system. A simple example of this is in Condor [11], where the system's scheduling policy protected light users were against starvation from heavy users.

Usability describes how well end-user actions are translated into resource bindings. This also describes the flexibility of the system, with respect to the granularity across time and space with which end-users can express their resource requests.

We have conducted initial experiments using data from PlanetLab workload traces and synthetic workloads. Of course, we expect to truly gain understanding of the system's behavior from experience in a live deployment with real users, particularly about the usability of the system.

Demand	s_0	s_1	d	$\{q_1..\}$	$\{r_1..\}$	v
Light	0	3	1	25	*any*	*all*
Heavy	0	3	1	50	*any*	*all*
				$\oplus$		
	0	3	1	25	*any*	$all \cdot \frac{1}{4}$

Table 2: Parameters for users in the experiment during periods of varying demand. We simulate heavy demand by doubling resource requests from all bids. *any* refers to any machine, and *all* refers to the entire balance of virtual currency. The two bids heavy-demand are clauses in an *xor* bid.

6.1 Experiments

We present an initial evaluation of Bellagio through a set of experiments designed to test the scalability and long-term behavior of the system.

6.1.1 Scalability

The main computational bottlenecks in Bellagio are the resource discovery algorithm in SWORD and the auction-clearing algorithm in SHARE. Both of these bottlenecks scale well with a reasonable number of users (bids) and resources.

Because the resource discovery algorithm in SWORD is distributed, it scales well to a large number of resources and client query rates. An analysis of SWORD indicates that the performance of the resource discovery algorithm requires less than 10 seconds on average to respond to a query for a query rate of 1800 queries per five minutes [13].

The performance of auction clearing depends on the auction-clearing algorithm being used. An analysis of the various heuristic algorithms shows that the greedy algorithm employed by Bellagio scales easily to a thousand bids and resources.

6.1.2 Efficiency

The following experiment illustrates how Bellagio handles during varying levels of demand.

The experiment consists of two auction-clearing periods for 120 resources. The first period is intended to simulate a system with light demand, and the second period simulates a system with heavy demand. Heavy demand is simulated by doubling the quantity of resources requested ($\{q_1\}$) in each bid. For simplicity, we assume that all resources are homogeneous and that all users express similar interests.

Given a situation with excess resource demand, we expect users to be able to tailor their applications to receive a smaller resource shares. We define five users for our experiment, each with equal amounts of virtual currency with preferences (bids) summarized in Table 2.

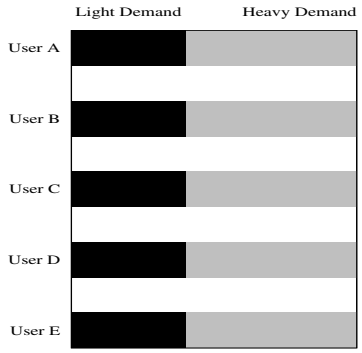

Figure 2: Utility derived by each user as a function of resource shares without Bellagio. Higher utility is represented by a dark-shaded box, and lighter shades represent less utility.

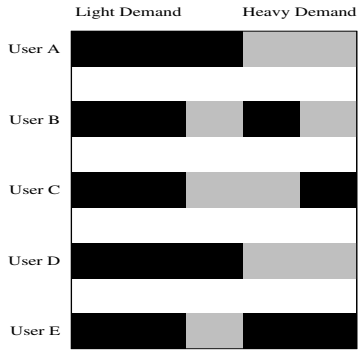

Figure 3: Utility derived by each user as a function of resource shares with Bellagio.

In Figures 2 and 3, we plot the utility derived from each of the five users based on the resources they receive. In this experiment, we define an application's utility by its bid amount. Higher utility is represented by dark boxes, and lower utility is represented by lighter boxes.

Figure 2 illustrates a scenario similar to Figure 1, where access to resources is controlled merely by some form of proportional share. In this scenario, each user receives her full allotment of resources during periods of light demand. However, when resource consumption increases, each user receives approximately 20 machines (resources). From Table 2, we see that they derive a utility of 0.25 for this resource set. Despite being able to tolerate the use of fewer resources, users have no incentive to slow down or delay resource consumption because they have no guarantee that other

users will also scale back usage. Therefore, each user receives this utility for the duration of excess demand.

Figure 3 illustrates how Bellagio handles resource contention. During period of light demand, Bellagio can satisfy all resource requests, as before. When resource contention increases, Bellagio can spread out resource requests in order to maximize the aggregate utility of the users. Bellagio assigns resources to each user that provides each user a time share in which she receives her full request of resources. This illustrates the potential benefit of allowing resource demand to be spread out over time to reduce the impact of resource contention.

6.1.3 Fairness

To measure the long-term behavior of resource distribution, we conduct an experiment with a set of five users with varying currency distributions, bidding for some portion of 100 resources. As before, the experiment consists of two periods: one of light demand, and one of heavy demand. At each auction-clearing period, all users submit a bid consisting of their entire balance of virtual currency. During the period of light demand, two users bid for some small set of resources that do not surpass the number of resources available in the system (same quantity as 2. During the period of heavy demand, bidders double the amount of resources requested. In addition, the three previously inactive bidders begin bidding during this period.

The results are in Figure 4.

During under-demand, only user 1 and user 5 are bidding, and since there is an under-demand for resources in this scenario, each bidder's resource request is completely satisfied.

For the latter time steps, there is contention for the resources. The bids are made using the entire balance of virtual currency, and for a duration of 1 time period. Initially the wealthier users are able to obtain more resources, but as the other users accrue more currency, they are eventually able to obtain their share of resources. As we can see from the figure, the distribution of resource winnings during periods of excess demand are proportional to the distribution of virtual currency. The distribution of resource winnings is less pronounced, however, for closer currency rates. We see this behavior in user 1 and user 2, who have similar currency rates.

As a result, Bellagio can use an appropriate currency distribution policy to protect light users against starvation from heavy users, as in Condor, but there is added flexibility in determining the actual proportion of resources being distributed to each user.

7 Future Work

Bellagio provides incentive for users to spread out resource demand in order to reduce some amount of resource contention. It exports an intuitive interface using a familiar economic model and hopefully will increase both the utilization of distributed resources as well as end-user utility. However, the performance of the system cannot be fully understood until a live deployment. We expect the method of resource reservation and details of the virtual economy to evolve as more end-users use the system.

As mentioned earlier, we plan to deploy this system on PlanetLab and gain experience from having real applications and workloads on the system. We are particularly interested in the extent to which the bidding language constrains the types of co-allocation that can be performed, and the effectiveness of virtual currency policies in spreading out resource demand over time.

Furthermore, we would like to be able to extend the model of a resource marketplace to a resource exchange. In this environment, users act as both producers and consumers of goods, where they can exchange resources or virtual currency to other users, and the auctioneer will simply act as the facilitator of exchanges to keep track of resource consumption and currency balance. The difficulty in implementing this is in the incentive design. As mentioned earlier, the payment rule in Bellagio provides incentive for users to bid truthfully for resources. In economic theory, this rule only applies to auctions and does not hold in economic exchanges.

References

[1] A. Alexandrov, M. Ibel, K. Schauser, , and C. Scheiman. Superweb: Research issues in java-based global computing. *Concurrency: Practice and Experience*, June 1997.

[2] A. Arpaci-Dusseau, D. Culler, and A. Mainwaring. Scheduling with implicit information in distributed systems. In *Proceedings of the 1998 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 233–243. ACM Press, 1998.

[3] C. Boutilier. Bidding languages for combinatorial auctions. In *In Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1211–1217, 2001.

[4] R. Buyya, D. Abramson, , and J. Giddy. Nimrodg: An architecture of a resource management and scheduling system in a global computational grid. In *In Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region*, pages 283–289, May 2000.

[5] D.C. Parkes C. Ng and M. Seltzer. Strategyproof Computing: Systems infrastructures for self-interested parties.

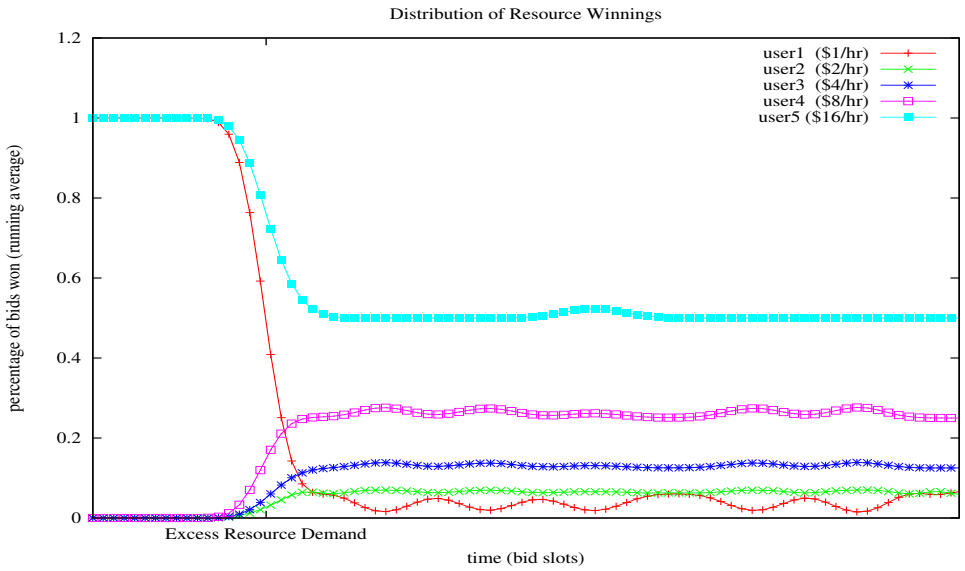

Figure 4: Percentage of resource requests satisfied for parties with different currency rates. The y-axis is expressed as a percentage of total resources won, taken as a weighted average over the previous 10 time slots. The point where excess-demand begins is labeled in the plot. The latter time-steps represent over-demand for system resources. The currency rate for each party is labeled in the plot.

In *1st Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[6] B. Chun, C. Ng, J. Albrecht, D. Parkes, and A. Vahdat. Computational resource exchanges for distributed resource allocation, 2004.

[7] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of SuperComputer Applications*, 15(3), 2001.

[8] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. Sharp: an architecture for secure resource peering. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, pages 133–148. ACM Press, 2003.

[9] K. Czajkowski and I. Foster and C. Kesselman and N. Karonis and S. Martin and W. Smith and and S. Tuecke. A resource management architecture for metacomputing systems. In *Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[10] K. Lai, B. Huberman, and L. Fine. Tycoon: A distributed market-based resource allocation system. Technical Report cs.DC/0404013, Hewlett Packard, 2004.

[11] M. Litzkow, M. Livny, , and M. Mutka. Condor - A Hunter of Idle Workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, pages 104–111, June 1988.

[12] N. Nisan. Bidding and allocation in combinatorial auctions. In *In Proceedings of the 2nd ACM Conference on Electronic Commerce*, pages 1–12, 2000.

[13] D. Oppenheimer, J. Albrecht, D. Patterson, and A. Vahdat. Scalable wide-area resource discovery. Technical Report UCB//SD-04-1334, UC Berkeley and UC San Diego, 2004.

[14] D. Parkes, J. Kalagnanam, and M. Eso. Achieving budget-balance with vickrey-based payment schemes in exchanges. In *Proceedings of Internation Joint Conference on Artificial Intelligence*, pages 1161–1168, 2001.

[15] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A blueprint for introducing disruptive technology into the internet, 2002.

[16] O. Regev and N. Nisan. The popcorn market - an online market for computational resources. In *Proceedings of the first international conference on Information and computation economies*, pages 148 – 157, 1998.

[17] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: A fast optimal algorithm for combinatorial auctions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1102–1108, 2001.

[18] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *In Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 69–76, 2002.

[19] I. Stoica, H. Abdel-Wahab, and A. Pothen. A microeconomic scheduler for parallel computers. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 200–218. Springer-Verlag, 1995.

[20] N. Stratford and R. Mortier. An economic approach to adaptive resource management. In *Workshop on Hot Topics in Operating Systems*, pages 142–147, 1999.

[21] C. Waldspurger, T. Hogg, B. Huberman, and J. Kephar. Spawn: A distributed computational economy. In *IEEE Transactions on Software Engineering*, pages 103 – 117, 1992.

[22] C. Waldspurger and E. Weihl. Stride scheduling: Deterministic proportional- share resource management. Technical report, Massachusetts Institute of Technology, 1995.

[23] R. Wolski, J. Plank, J. Brevik, and T. Bryan. Analyzing market-based resource allocation strategies for the computational grid, 2001.

[24] E. Zurel and N. Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 125–136. ACM Press, 2001.