G-commerce: Market Formulations Controlling Resource Allocation on the Computational Grid*

Rich Wolski, James S. Plank, Todd Bryan
Computer Science Department
University of Tennessee

John Brevik
Mathematics and Computer Science Department
College of the Holy Cross

Abstract

In this paper, we investigate G-commerce — computational economies for controlling resource allocation in Computational Grid settings. We define hypothetical resource consumers (representing users and Grid-aware applications) and resource producers (representing resource owners who "sell" their resources to the Grid). We then measure the efficiency of resource allocation under two different market conditions: commodities markets and auctions. We compare both market strategies in terms of price stability, market equilibrium, consumer efficiency, and producer efficiency. Our results indicate that commodities markets are a better choice for controlling Grid resources than previously defined auction strategies.

1. Introduction

A new computing paradigm known as the Computational Grid [13, 3] articulates a vision of distributed computing in which applications "plug" into a "power grid" of computational resources when they execute, dynamically drawing what they need from the global supply. While a great deal of research concerning the software mechanisms that will be necessary to bring Computational Grids to fruition is underway [3, 12, 16, 7, 21], little work has focused on the resource control policies that are likely to succeed. Almost all Grid resource allocation and scheduling research espouses one of two paradigms: centralized omnipotent resource control [14, 16, 18] or localized application control [8, 4, 2, 15]. The first is certainly not a scalable solution and the second can clearly lead to unstable resource assignments as "Grid-aware" applications adapt to compete for resources.

In this paper, we investigate **G-commerce** – a method of dynamic Grid resource allocation built on the notion of a market economy. Framing the resource allocation problem in economic terms is attractive for several reasons. First, resource usage is not free. Second, the dynamics of Grid performance response are, as of yet, difficult to model. As resource load fluctuates, applications can adjust their resource usage at machine speeds, forming a feedback control loop with a potentially non-linear response. By formulating Grid resource usage in market terms, we are able to draw upon a large body of analytical research from the field of economics and apply it to the understanding of emergent Grid behavior. Last, if resource owners are to be convinced to federate their resources to the Grid, they must be able to account for the relative costs and benefits of doing so.

We focus on two broad categories of G-commerce formulations: **commodities markets** and **auctions**. Modeling the Grid as a commodities market is natural since the Grid strives to allow applications to treat disparate resources as interchangeable commodities. On the other hand, auctions require little in the way of global price information, and they are easy to implement in a distributed setting. Both types of economies have been studied as strategies for distributed resource brokering [10, 22, 17, 5, 6, 9]. Our goal is to enhance our deeper understanding of how these economies will fare as resource brokering mechanisms for Computational Grids.

We evaluate commodities markets and auctions with respect to four criteria: Grid-wide price stability, market equilibrium, application efficiency, and resource efficiency. Price stability is critical to ensure scheduling stability. If the price fluctuates wildly, application and resource schedulers that base their decisions on the state of the economy will follow suit, again leading to poor performance, and therefore ineffectiveness of the Grid as a computational infrastructure. Equilibrium measures the degree to which prices are fair. If the overall market cannot be brought into equilibrium, the relative expense or worth of a particular transaction cannot be trusted, and again the Grid is not doing its job. Application efficiency measures how effective the Grid is as a computational platform, and resource efficiency measures how well the Grid manages its resources. Poor ap-

*This work was supported, in part, by NSF grants EIA-9975020, EIA-9975015, ACI-9876895.

plication and/or resource efficiency will mean that the Grid is not succeeding as a computational infrastructure.

The remainder of this paper is organized as follows. In the next section, we discuss the specific market formulations we use in this study. Section 3 describes the simulation methodology we use and the results we obtain for different hypothetical market parameterizations. In Section 4 we conclude and point to future work.

2. G-commerce — Market Economies for the Grid

In formulating a computational economy for the Grid, we assume that the following premise must be true: *The relative worth of a resource is determined by its supply and the demand for it.* This assumption is important because it rules out pricing schemes that are based on arbitrarily decided priorities. While there are several plausible scenarios in which such Draconian policies are appropriate (e.g. users are funded to use a specific machine as part of their individual research projects), from the perspective of the Grid, the resource allocation problem under these conditions has been solved.

Next, we assume that relative worth, and not price, is determined by supply and demand. Supply and demand are functions of price, and relative worth is determined by some optimization function over the space of prices. For example, in this paper, we will consider the price to be representative of relative worth at the price-point that equalizes supply and demand – that is, at market equilibrium. Conversely, at a non-equilibrium price-point (where supply does not equal demand), price either overstates or understates relative worth.

Another important aspect of our approach is that we do not restrict the definition of currency or the rules governing its supply. If users or applications are given currency from outside the system, we would expect inflationary price behavior, but the market will remain intact. Also, it is possible to segregate computational consumers and producers. In a "true" market, producers are expected to spend their profits (somewhere) within the economy eventually. While we believe our results remain valid for this more restricted case, in this work we model producers and consumers as disjoint entities.

Finally, and most pragmatically, if we are to simulate a computational economy we must ultimately hypothesize supply and demand functions for our simulated producers and consumers respectively. Individual supply and demand functions are difficult to measure at best, particularly since there are no existing Computational Grid economies at present. Our admittedly less satisfactory approach is to define supply and demand functions that represent each simulated producer and consumer's "self-interest." An individual consumer buys only if the purchase is a "good deal" for that consumer. Analogously, producers sell only when a sale is in their best interest. We believe resource decisions based on self-interest are inescapable in any federated resource system.

2.1. Producer Models

We simulate two different commodity producers in this study: CPU and disk storage. This set of simulated producers is used to compare commodity and auction-based market settings. While the results should generalize to include a variety of other commodities, networks present a special problem. For the moment we ignore networks and instead allow CPU and disk requests to be serviced by any provider, regardless of network connectivity.

In this study, a CPU represents a computational engine with a fixed speed. A CPU producer agrees to sell to the Grid some number of fixed "slots" of the CPU it controls. The number of slots depends on the speed of the producer's CPU (a faster CPU will provide more slots than a slower one), and what fraction of the CPU the producer wishes to supply to the Grid. When a job occupies a CPU, it is guaranteed to get whatever percentage of the CPU will provide the dedicated speed. Each CPU producer differs in the total number of slots it is willing to sell to the Grid.

To determine supply at a given price-point, each CPU calculates

$$mean\_price = (revenue/elapsed\_time)/slots \quad (1)$$

where $revenue$ is the total amount of Grid currency (hereafter referred to as \$G which is pronounced "Grid bucks"), $elapsed\_time$ is an incrementing clock, and $slots$ is the total number of process slots the CPU owner is willing to support. The $mean\_price$ value is the average \$G per time unit per slot the CPU has made from selling to the Grid. In our study, CPU producers will only sell if the current price of a CPU slot exceeds the $mean\_price$ value, and when they sell, they sell all unoccupied slots. That is, the CPU will sell all of its available slots when it will turn a profit (per slot) with respect to the average profit over time.

The model we use for a disk producer is similar to that for the CPU producer, except that disks sell some number of fixed-sized "files" that applications may use for storage. The $mean\_price$ calculation for disk files is

$$mean\_price = (revenue/elapsed\_time)/capacity \quad (2)$$

where $capacity$ is the total number of files a disk producer is willing to sell to the Grid. If the current price for a file is greater than the $mean\_price$, a disk producer will sell all of its available files.

Note that the resolution of CPU slots and file sizes is variable. Since our markets transact business at the com-

modity level, however, we hypothesize that any real implementation for the Grid will need to work with larger-scale aggregations of resources for reasons of efficiency. For the simulations described in Section 3 we choose values for these aggregations that we believe reflect a market formulation that is currently implementable.

2.2. Consumers and Jobs

Consumers express their needs to the market in the form of jobs. Each job specifies both a size and an occupancy duration for each resource to be consumed. Each consumer also sports a budget of $G that it can use to pay for the resources needed by its jobs. Consumers are given an initial budget and a periodic allowance, but they are not allowed to hold $G over from one period until the next. This method of budget refresh is inspired by the allocation policies currently in use at the NSF Partnerships for Advanced Computational Infrastructure (PACIs), where allocations are perishable.

When a consumer wishes to purchase resources for a job, it declares the size of the request for each commodity, but not the duration. At the time a producer agrees to sell to a consumer, a price is fixed that will be charged to the consumer for each simulated time unit until the job completes. Consider in example a consumer wishing to buy a CPU slot for 100 simulated minutes and a disk file for 300 simulated minutes to service a particular job. If the consumer wishes to buy each for a particular price, it declares to the market a demand of 1 CPU slot and 1 disk slot, but does not reveal the 100 and 300 minute durations. A CPU producer wishing to sell at the CPU price agrees to accept the job until either the job completes or the consumer's budget goes to zero (as does the disk producer for the disk job). Once the sales are transacted, the consumer's budget is decremented by the agreed-upon price every simulated minute, and each producer's revenue account is incremented by the same amount. If the job completes, the CPU producer will have accrued 100 times the CPU price, the disk producer will have accrued 300 times the disk price, and the consumer's budget will have been decremented by the sum of 100 times the CPU price and 300 times the disk price.

In defining this method of conducting resource transactions, we make several assumptions. First, we assume that in an actual Grid setting resource producers or suppliers will commit some fraction of their resource to the Grid, and that fraction is slowly changing. Once committed, the fraction "belongs" to the Grid so producers are not concerned with occupancy. We are also assuming that neither consumers nor producers are malicious and that both honor their commitments. In practice, this requirement might be satisfied with secure authentication methods and libraries.

The consumer demand function is somewhat more complex than the CPU and disk supply functions. Consumers must purchase enough CPU and disk resource for each job they wish to run. If they cannot afford the request for only one type, they do not express demand for the other. That is, the demand functions for CPU and disks are strongly correlated, but the supply functions are not. This relationship between supply and demand functions constitutes the most difficult of market conditions. Most market systems make weaker assumptions about the difference in correlation. By addressing the more difficult case, we believe our work more closely resembles what can be realized in practice.

To determine their demand at a given price, each consumer first calculates the average rate at which it would have spent $G for the jobs it has run so far if it had been charged the current price. It then computes how many $G it can spend per simulated time unit until the next budget refresh. That is, it computes

$$avg\_rate = \frac{\sum total\_work_i * price_i i}{now} \quad (3)$$

$$capable\_rate = \frac{remaining\_budget}{(refresh - now)} \quad (4)$$

where $total\_work_i$ is the total amount of work performed so far using commodity i, $price_i$ is the current price for commodity i, $remaining\_budget$ is the amount left to spend before the budget refresh, $refresh$ is the budget refresh time, and now is the current time. When $capable\_rate$ is greater than or equal to avg_rate, a consumer will express demand. Note that the demand function does not consider past price performance. Rather, consumers act opportunistically based on the amount of money left to spend before a budget refresh.

Consumers, in our simulations, generate work as a function of time. We arbitrarily fix some simulated period to be a "simulated day." At the beginning of each day, every consumer generates a random number of jobs. By doing so, we hope to model the diurnal user behavior that is typical in large-scale computational settings. In addition, each consumer can generate a single new job every time step with a pre-determined probability. Consumers maintain a queue of jobs waiting for service before they are accepted by producers. When calculating demand, they compute avg_rate and $capable\_rate$ and demand as many jobs from this queue as they can afford.

To summarize, for our G-commerce simulations,

- all entities except the market-maker act individually in their respective self-interests,
- producers consider long-term profit and past performance when deciding to sell,
- consumers are given periodic budget replenishments and spend opportunistically, and

- consumers introduce work loads in bulk at the beginning of each simulated day, and randomly throughout the day.

We believe that this combination of characteristics captures likely producer and consumer traits in real Grid settings.

2.3. Commodities Markets and Dynamic Pricing

To implement a market economy, we require a pricing methodology that produces a system of price adjustments which bring about market equilibrium (i.e. equalizes supply and demand). From a theoretical standpoint, a *market economy* is a system involving producers, consumers, several commodities, and supply and demand functions for each commodity which are determined by the set of market prices for the various commodities [11]. A unique equilibrium price is guaranteed to exist in this framework by a theorem of Debreu ([11], Chapter 5), the proof of which is non-constructive and involves topological methods. In [20], Smale produced a means for proving the existence of equilibrium which also entails a scheme for price adjustments to reach it.

If commodity prices are represented as a *price vector* $\mathbf{p} = \begin{pmatrix} p_1 & p_2 & \dots & p_n \end{pmatrix}$, where p_i stands for the price of the i^{th} commodity, we can define *excess demand* z_j for the j^{th} commodity as the demand minus the supply. As defined, z_j may be positive or negative; negative excess demand can be interpreted simply as excess supply. We assume that the markets for these commodities may be interrelated, so that each z_j is a function of all of the prices p_i, that is, of the vector $\mathbf{p}$. Smale's theorem says given a market consisting of n interrelated commodities with price vector $\mathbf{p}$ and associated excess demand vector $\mathbf{z}(\mathbf{p}) = \mathbf{z} =$ an *equilibrium point* with $\mathbf{p}^*$ such that $\mathbf{z}(\mathbf{p}^*) = 0$ exists [24]. Moreover, for any value of $\mathbf{p}$, we can form the $n \times n$ matrix of partial derivatives

$$D_{\mathbf{z}}(\mathbf{p}) = \left(\frac{\partial z_i}{\partial p_j} \right).$$

Then for any value of λ which has the same sign as the determinant of $D_{\mathbf{z}}(\mathbf{p})$, we can obtain economic equilibrium by always obeying the differential equation

$$D_{\mathbf{z}}(\mathbf{p}) \frac{d\mathbf{p}}{dt} = -\lambda \mathbf{z}(\mathbf{p}). \tag{5}$$

Observe that taking $\lambda = 1$ and applying the Euler discretization at positive integer values of t reduces this process to the Newton-Raphson method for solving $\mathbf{z}(\mathbf{p}) = \mathbf{0}$; for this reason, Smale refers to this process as "global Newton."

Obtaining the partial derivatives necessary to carry out Smale's process in an actual economy is impossible; however, within the framework of our simulated economy, we are able to get good approximations for the partials at a given price vector by polling the producers and consumers. We will refer, conveniently but somewhat inaccurately, to this price adjustment scheme as *Smale's method*. The obvious drawback to the such a scheme is that it relies on polling aggregate supply and demand repeatedly to obtain the partial derivatives of the excess demand functions. In practice, we do not wish to assume that such polling information will be available.

A theoretically attractive way to circumvent this difficulty is to approximate each excess demand function z_i by a polynomial in $p_1, p_2, ..., p_n$ which fits recent price and excess demand vectors and to use the partial derivatives of these polynomials in Equation 5. In simulations, this method does not, in general, produce prices which approach equilibrium. The *First Bank of G* is a price adjustment scheme which both is practicable and gives good results; this scheme involves using *tâtonnement* [23] until prices get "close" to equilibrium, in the sense that excess demands have sufficiently small absolute value, and then using the polynomial method for "fine tuning." Thus, the First Bank of G approximates Smale's method but is implementable in real-world Grid settings since it hypothesizes excess demand functions and need not poll the market for them. Our experience is that fairly high-degree polynomials are required to capture excess demand behavior with the sharp discontinuities described above. For all simulations described in Section 3, we use a degree 17 polynomial.

2.4. Auctions

Auctions have been extensively studied as resource allocation strategies for distributed computing systems. In a typical auction system (e.g. [10, 22, 17, 5]), resource producers (typically CPU producers) auction themselves using a centralized auctioneer and sealed-bid, second-price auctions. This and other auction variants are described in [5].

When consumers simply desire one commodity, for example CPUs in Popcorn [17], auctions provide a convenient, straightforward mechanism for clearing the marketplace. But when an application (the consumer in a Grid Computing scenario) desires multiple commodities, it must place simultaneous bids in multiple auctions, and may only be successful in a few of these. When this happens, it must expend currency on the resources that it has obtained while it waits to obtain the others.

Also, while a commodities market presents an application with a resource's worth in terms of its price, thus allowing the application to make meaningful scheduling decisions, an auction is more unreliable in terms of both pricing and the ability to obtain a resource, and may therefore result in poor scheduling decisions and more inefficiency for consumers.

To gain a better understanding of how auctions fare in comparison to commodities markets, we implement the following simulation of an auction-based resource allocation mechanism for computational grids. At each time step, CPU and disk producers submit their unused CPU and file slots to a CPU and a disk auctioneer. These are accompanied by a minimum selling price, which is the average profit per slot, as detailed in Section 2.1 above. Consumers use the demand function as described in Section 2.2 to define their bid prices, and as long as they have money to bid on a job, and a job for which to bid, they bid on each commodity needed by their oldest uncommenced job.

Once the auctioneers have received all bids for a time step, they cycle through all the commodities in a random order, performing one second-price auction per commodity. In each auction, the highest-bidding consumer gets the commodity if the bid price is greater than the commodity's minimum price. If there is no second-highest bidder, then the price of the commodity is the average of the commodity's minimum selling price and the consumer's bid price. Auctions are transacted in this manner for every commodity, and this process is repeated at every time step.

Note our structuring of the auctions requires that each consumer can have at most one job for which it is currently bidding. When it obtains all the resources for that job, it immediately starts bidding on its next job. When a time step expires and all auctions for that time step have been completed, there may be several consumers whose jobs have some resources allocated and some unallocated, as a result of failed bidding. These consumers have to pay for their allocated resources while they wait to start bidding in the next time step.

While the auctions determine transaction prices based on individual bids, the supply and demand functions used by the producers and consumers to set ask and bid prices are the same functions we use in the commodities market formulations. Thus, we can compare the market behavior and individual producer and consumer behavior in both auction and commodity market settings.

3. Simulations and Results

We compare commodities markets and auctions using the producers and consumers described in Section 2.1 in two overall market settings. In the first, which we term *under-demand*, producers are on average capable of supporting enough demand to service all of the jobs consumers can afford. Recall that our markets do not include resale components. Consumers do not make money. Instead, \$G are given to them periodically much the in the same way that PACIs dole out machine-time allocations. Similarly, producers do not spend money. Once gathered, it is hoarded. The under-demand case corresponds to a working Grid economy in which the allocations correctly match the available resources. That is, when the rate that \$G are allocated to consumers roughly matches the rate at which they introduce work to the Grid. In the *over-demand* case, consumers wish to buy more resource than is available. That is, they generate work fast enough to keep all producers almost completely busy thereby creating a work back-log. Table 1 completely describes the invariant simulation parameters we chose for both cases. For the under-demand simulation, we defined 100 consumers to use the 100 CPUs and disks, where each consumer submitted a random number of jobs (between 1 and 100) at every day-break, and had a 10% chance of submitting a new job every time unit. The over-demand simulation specified 500 of the same consumers, with all other parameters held constant.

CPUs	100
disks	100
CPU slots per CPU	[2 .. 10]
disk files per disk	[1 .. 15]
CPU job length	[1 .. 60] time units
disk job length	[1 .. 60] time units
simulated day	1440 time units
allowance period	[1 .. 10] days
jobs submitted at day-break	[1 .. 100]
new job probability	10%
allowance	10^6 \$G
Bank of G Polynomial Degree	17
λ factor	.01

Table 1. Invariant simulation parameters for this study

Using our simulated markets, we wish to investigate three questions with respect to commodities markets and auctions: Do the theoretical results from Smale's work [19] apply to plausible Grid simulations? Can we approximate Smale's method with one which is practically implementable? Are auctions or commodities markets a better choice for Grid computational economies?

If Smale's results apply, they dictate that an equilibrium price-point must exist (in a commodity market formulation), and they provide a methodology for finding those prices that make up the price-point. Assuming Smale's results apply, we also wish to explore methodologies that achieve Smale's results, but which are implementable in real Grid settings. Lastly, recent work in Grid economies [1, 14, 18] and much previous work in computational economic settings [10, 17, 5, 22] has centered on auctions as the appropriate market formulation. We wish to determine whether commodities markets are a better alternative to auctions.

3.1. Market Conditions

Figure 1 shows the CPU and disk prices for Smale's method in our simulated Grid economy over $10,000$ time units. The diurnal nature of consumer job submission is

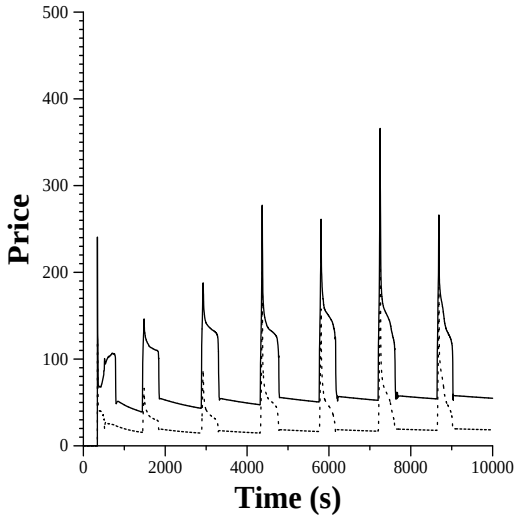


Figure 1. Smale's prices for the under-demand case. Solid line is CPU price, and dotted line is disk price in $G

evident from the price fluctuations. Every 1440 "minutes" each consumer generates between 1 and 100 new jobs causing demand and prices to spike. However, Smale's method is able to find an equilibrium price for both commodities quickly, as is evidenced in Figure 2. Notice that the ex-

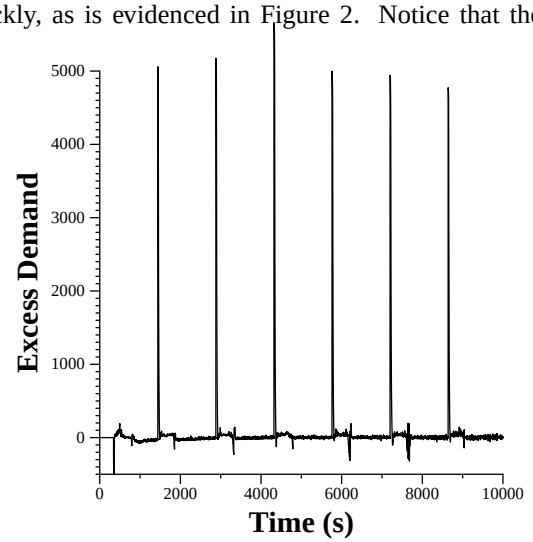


Figure 2. Smale's CPU excess demand for the under-demand case. The units are CPU slots.

cess demand spikes in conjunction with the diurnal load, but is quickly brought to zero by the pricing shown in Figure 1 where is hovers until the next cycle. Disk excess demand is similar and is thus omitted for brevity. Again, market equilibrium is quickly achieved despite the cyclic and non-smooth aggregate supply and demand functions implemented by the producers and consumers.

In Figure 3 we show the pricing determined by our engineering approximation to Smale's method — the First Bank of G (see Section 2.3 for details). The First Bank of G pric-

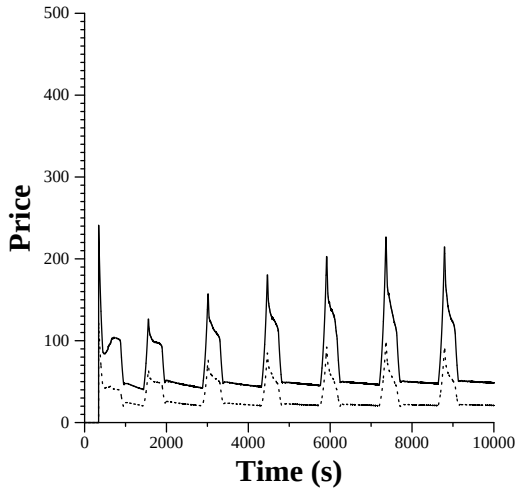


Figure 3. First Bank of G prices for the under-demand case. Solid line is CPU price, and dotted line is disk price in $G

ing closely approximates the theoretically achievable results generated by Smale's method in our simulated environment. Figure 4 shows CPU excess demand measures generated by First Bank of G pricing over the simulated period. While the excess demands for both commodities are not as tightly controlled as with Smale's method, the First Bank of G keeps prices very near equilibrium.

The pricing determined by auctions is quite different, however, as depicted in Figure 5 (we show only CPU price as disk price is almost identical). In the figure, we show the average price paid by all consumers for CPU during each auction round. We use the average price for all auctions as being representative of the "global" market price. Even though this price is smoothed as an average (some consumers pay more and some pay less during each time step), it shows considerably more variance than the commodity market set prices. The spikes in workload are not reflected in the price, and the variance seems to increase (i.e. the price becomes less stable) over time. Furthermore, disk pricing (not shown) is virtually identical. Disk resources are more plentiful in our simulations so disk prices should

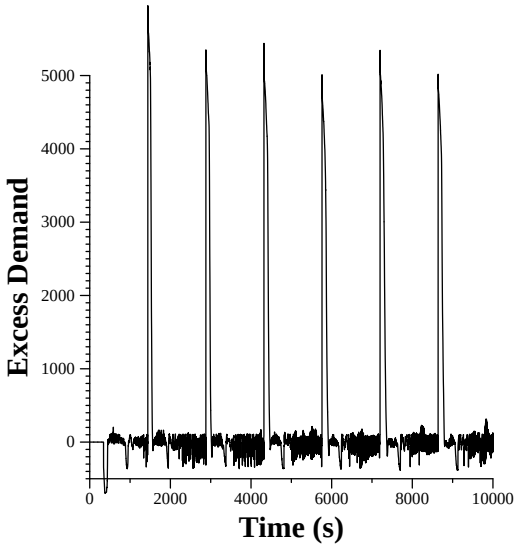
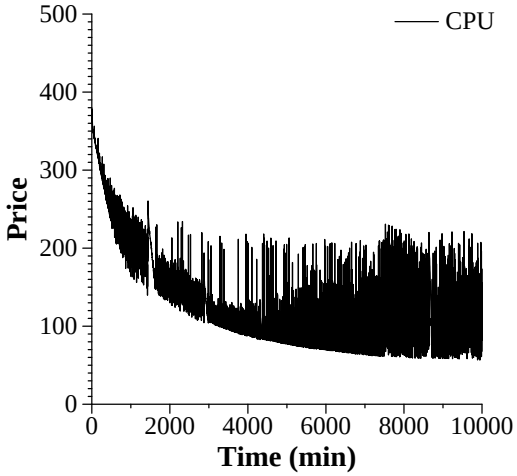


Figure 4. First Bank of G CPU excess demand for the under-demand case. The units are CPU slots.



Using Auctions

Figure 5. Auction prices for the under-demand case, average CPU price only, in $G

be lower in a healthy economy. The auction fails to capture this relationship, but the commodities market (both theoretically and practically) correctly determines a higher price for the scarce resource.

Excess demand for an auction is more difficult to measure since prices are negotiated between individual buyers and sellers. As an approximation, we consider the sum of unsatisfied bids and the number of auctions that did not make a sale as a measure of market equilibrium. The absolute value of this measure fulfills our notion of *absolute excess demand* for an auction. In terms of absolute excess demand, auctions never set prices which satisfy the market. For space reasons our results cannot be presented here, but they can be viewed in full elsewhere [24].

From these graphs we conclude that Smale's method is appropriate for modeling hypothetical Grid market and that the First Bank of G is a reasonable (and implementable) approximation of this method. These results are somewhat surprising given the discrete and sharply changing supply and demand functions used by our producers and consumers. Smale's proofs assume continuous functions and readily available partial derivatives. We also note that auctioneering, while attractive from an implementation standpoint, does not produce stable pricing or market equilibrium. If Grid resource allocation decisions are based on auctions, they will share this instability and lack of fairness. Conversely, a commodities market formulation, at least in simulation, performs better *from the standpoint of the Grid as a whole*. These results agree with those reported in [22] which indicate that auctions are locally advantageous, but may exhibit volatile emergent behavior system wide.

For the over-demanded market case, we increased the number of consumers to 500 leaving all other parameters fixed. The results were similar prompting us to omit their bulk, although the full results are available [24]. For our current purposes we will assert that as in the underdemand case, Smale's method provided price stability and tight control of excess demand, while the First Bank of G closely approximated Smale's method. Auctions generated unstable price series and provided poor control of excess demand.

3.2. Efficiency

While commodities markets using Smale's method of price determination appear to offer better theoretical and simulated economic properties (equilibrium and price stability) than auctions do, we also wish to consider the effect of the two pricing schemes on producer and consumer efficiency. To do so, we report the average percentage of time each resource is occupied as a utilization metric for suppliers, and the average number of jobs/minute each consumer was able to complete as a consumer metric. Table 2 summarizes these values for both the over- and under-demand cases. In terms of efficiency, Smale's method is best and the First Bank of G achieves almost the same results. Both are significantly better than the auction in all metrics except disk utilization in the over-demanded case. In general, however, Smale's method and the First Bank of G approximation both outperform the auction in the simulated Grid setting.

efficiency metric	under-demand	over-demand
Smale **consumer jobs/min**	0.14 j/m	0.05 j/m
B of G **consumer jobs/min**	0.13 j/m	0.04 j/m
auction consumer jobs/min	0.07 j/m	0.03 j/m
Smale **CPU** utilization %	60.7%	98.2%
B of G **CPU** utilization %	60.4%	93.9%
auction **CPU** utilization %	35.2%	85.5%
Smale **disk** utilization %	54.7%	88.3%
B of G **disk** utilization %	54.3%	84.6%
auction **disk** utilization %	37.6%	85.1%

Table 2. Consumer and Producer efficiencies

4. Conclusions and Future Work

We investigate two market strategies for setting prices in a computational economy: commodities markets and auctions. Commodities markets are a natural choice given the fundamental tenets of the Grid [13]. Auctions, however, are simple to implement and widely studied. We are interested in which methodology is most appropriate for Grid settings. To investigate this question, we examine the overall price stability, market equilibrium, producer efficiency, and consumer efficiency achieved by three methods in simulation. Our results show that Smale's results hold for our simulated Grid environment, despite badly behaved excess demand functions, and that the First Bank of G achieves results only slightly less desirable. In all cases, auctions are an inferior choice.

References

[1] D. Abramson, J. Giddy, I. Foster, and L. Kotler. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid ? In *Proceedings of the International Parallel and Distributed Processing Symposium*, May 2000.

[2] O. Arndt, B. Freisleben, T. Kielmann, and F. Thilo. Scheduling parallel applications in networks of mixed uniprocessor/multiprocessor workstations. In *Proceedings of ISCA 11th Conference on Parallel and Distributed Computing*, September 1998.

[3] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, L. J. Dennis Gannon, K. Kennedy, C. Kesselman, D. Reed, L. Torczon, , and R. Wolski. The grads project: Software support for high-level grid application development. Technical Report Rice COMPTR00-355, Rice University, February 2000.

[4] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing 1996*, 1996.

[5] J. Bredin, D. Kotz, and D. Rus. Market-based resource control for mobile agents. In *Second International Conference on Autonomous Agents*, pages 197–204. ACM Press, May 1998.

[6] J. Bredin, D. Kotz, and D. Rus. Utility driven mobile-agent scheduling. Technical Report PCS-TR98-331, Dartmouth College, Computer Science, Hanover, NH, October 1998.

[7] H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 1997.

[8] H. Casanova, G. Obertelli, F. Berman, and R. Wolski. The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid. In *Proceedings of SC00*, November 2000.

[9] J. Q. Cheng and M. P. Wellman. The WALRAS algorithm: A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24, 1998.

[10] B. Chun and D. E. Culler. Market-based proportional resource sharing for clusters. Millenium Project Research Report, `http://www.cs.berkeley.edu/~bnc/papers/market.pdf`, Sep 1999.

[11] G. Debreu. *Theory of Value*. Yale University Press, 1959.

[12] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 1997.

[13] I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.

[14] I. Foster, A. Roy, and L. Winkler. A quality of service architecture that combines resource reservation and application adaptation. In *Proceedings of TERENA Networking Conference*, 2000. to appear.

[15] J. Gehrinf and A. Reinfeld. Mars - a framework for minimizing the job execution time in a metacomputing environment. *Proceedings of Future general Computer Systems*, 1996.

[16] A. S. Grimshaw, W. A. Wulf, J. C. French, A. C. Weaver, and P. F. Reynolds. Legion: The next logical step toward a nationwide virtual computer. Technical Report CS-94-21, University of Virginia, 1994.

[17] N. Nisan, S. London, O. Regev, and N. Camiel. Globally distributed computation over the Internet — the POPCORN project. In *International Conference on Distributed Computing Systems*, 1998.

[18] B. Rajkumar. economygrid home page `http://www.computingportals.org/projects/economyManager.xml.html`.

[19] S. Smale. Dynamics in general equilibrium theory. *American Economic Review*, 66(2):284–294, May 1976.

[20] S. Smale. Convergent process of price adjustment and global newton methods. *Contributions to Economic Analysis*, 105:191–205, 1977.

[21] T. Tannenbaum and M. Litzkow. The condor distributed processing system. *Dr. Dobbs Journal*, February 1995.

[22] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A distributed computational economy. *IEEE Trans. on Software Engineering*, 18(2):103–117, February 1992.

[23] L. Walras. *Elements of pure economics; or, The theory of social wealth*. Allen and Unwin, 1954.

[24] R. Wolski, J. Plank, J. Brevik, and T. Bryan. G-commerce – market formulations controlling resource allocation on the computational grid. Technical Report UT-CS-00-450, University of Tennessee, October 2000.