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

Rich Wolski† James S. Plank† John Brevik‡ Todd Bryan†

† Department of Computer Science
University of Tennessee

‡ Mathematics and Computer Science Department
College of the Holy Cross

University of Tennessee Technical Report UT-CS-00-450
http://www.cs.utk.edu/~rich/publications/CS-00-450.ps.gz

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.

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

1 Introduction

With the proliferation of the Internet comes the possibility of aggregating vast collections of computers into large-scale computational platforms. A new computing paradigm known as the Computational Grid [15, 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, 14, 18, 8, 4, 20, 19, 1, 28], little work has focused on the resource control policies that are likely to succeed. In particular, almost all Grid resource allocation and scheduling and research espouses one of two paradigms: centralized omnipotent resource control [16, 18, 24, 25] or localized application control [9, 4, 2, 17]. The first is certainly not a scalable solution and the second can lead to unstable resource assignments as "Grid-aware" applications adapt to compete for resources.

In this paper, we investigate **G-commerce** — the problem of dynamic resource allocation on the Grid in terms of computational *market economies* in which applications must buy the resources they use from resource suppliers using an agreed-upon currency. Framing the resource allocation problem in economic terms is attractive for several reasons. First, resource usage is not free. While burgeoning Grid systems are willing to make resources readily available to early developers as a way of cultivating a user community, resource cost eventually must be considered if the Grid is to become pervasive. Second, the dynamics of Grid performance response are, as of yet, difficult to model. Application schedulers can make resource acquisition decisions at machine speeds in response to the perceived effects of contention. As resource load fluctuates, applications can adjust their resource usage, 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. Any market formulation carries with it an inherent notion of relative worth which can be used to quantify the cost-to-benefit ratio for both Grid users and stake-holders.

While there are a number of different plausible G-commerce market formulations for the Grid, we focus on two broad categories: **commodities markets** and **auctions**. The overall goal of the Computational Grid is to allow applications to treat computational, network, and storage resources as individual and interchangeable commodities, and not specific machines, networks, and disk or tape systems. Modeling the Grid as a commodities market is thus a natural choice. 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 [11, 29, 21, 6, 7, 10]. Our goal is to enhance our deeper understanding how these economies will fare as resource brokering mechanisms for Computational Grids.

To investigate Computational Grid settings and G-commerce resource allocation strategies, we evaluate commodities markets and auctions with respect to four criteria:

1. Grid-wide price stability
2. Market equilibrium
3. Application efficiency
4. 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, 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. Resource efficiency measures how well the Grid manages its resources. Poor application and/or resource efficiency will mean that the Grid is not succeeding as a computational infrastructure. Thus, we use these four criteria to evaluate how well each G-commerce economy works as the basis for resource allocation in Computational Grids.

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 make four assumptions. *#1: 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. For example, it is not possible in an economy for an organization to simply declare what the price of its resources are and then decree that its users pay that price even if cheaper, better alternatives are available. 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.

#2: Relative worth, and not price, are 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.

#3: 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.

#4: Resource decisions based on self-interest are inescapable in any federated resource system. 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.

In the next section, Section 2.1 we detail the specific functions we investigate, but generally our approach assumes that individuals act only in their own self-interest.

2.1 Producers and Consumers

To compare the efficacy of commodities markets and auctions as Grid resource allocation schemes, we define a set of simulated Grid resource producers and consumers representing resource providers and applications respectively. We then use the same set of producers and consumers to compare commodity and auction-based market settings.

We simulate two different commodity producers in this study: CPU and disk storage. That is, from the perspective of a resource market, there are two kinds of resources within our simulated Grids: CPUs and disks. While the results should generalize to include a variety other commodities, networks present a special problem. Our consumer model is that an application may request a specified amount of CPU and disk (the units of which we discuss below) and that these requests may be serviced by any provider regardless of location or network connectivity. Since network links cannot be combined with other resources arbitrarily, they cannot be modeled as separate commodities. We believe that network cost can be represented in terms of "shipping" costs in more complicated markets, but for the purposes of this study, we consider network connectivity to be uniform.

2.1.1 CPU Producer Model

In this study, a CPU represents a computational engine with a fixed dedicated speed. A CPU producer agrees to sell to the Grid some number of fixed "shares" of the CPU it controls. The real-world scenario for this model is for CPU owners to agree to host a fixed number of processes from the Grid in exchange for Grid currency. Each process gets a fixed, pre-determined fraction of the dedicated CPU speed, but the owner determines how many fractions or "slots" he or she is willing to sell. For example, in our study, the fraction is 10% so each CPU producer agrees to sell a fixed number (less than 10) of 10%-sized slots to the Grid. When a job occupies a CPU, it is guaranteed to get 10% of the available cycles for each slot it consumes. Each CPU, however, differs in the total number of slots it is willing to sell.

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

$$mean\_price = revenue/now/slots \qquad (1)$$

where $revenue$ is the total amount of Grid currency (heretofore to be referred to as \$G which is pronounced "Grid bucks"), now 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 with it will turn a profit (per slot) with respect to the average profit over time.

2.1.2 Disk Producer Model

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

$$mean\_price = revenue/now/capacity \qquad (2)$$

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. It is possible to make a CPU slot span the duration of a single clock cycle, and a disk file be a single byte. Since our markets transact business at the commodity 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.

2.1.3 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). At these centers, 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. Our model is that job durations are relatively long, and that producers allow consumers occupancy without knowing for how long the occupancy will last. 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.

For example, consider a consumer wishing to buy a CPU slot for 100 minutes and a disk file for 300 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 the job completes (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 resources to the Grid, and that fraction is slowly changing. Once committed, the fraction "belongs" to the Grid so producers are not concerned with occupancy. They are concerned, in our models, with profit and they only sell if it is profitable on the average. By including time in the supply functions, producers consider past occupancy (in terms of profit) when deciding to sell. We are also assuming that neither consumers nor producers are malicious and that both honor their commitments. In practice, this requirement assuredly may be difficult to enforce. If consumers and producers must agree to use secure authentication methods and system provided libraries to gain access to Grid resources, then it should be possible.

2.1.4 Consumer Demand

The consumer demand function is more complex than the CPU and disk supply functions. Consumers must purchase enough CPU and disk resources for each job they wish to run. If they cannot satisfy 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 (although 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

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

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

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.

Unlike our supply functions, the consumer demand function does not consider past price performance directly when determining demand. Instead, consumers using this function act opportunistically based on the money they have left to spend and when they will receive more. They use past behavior only as an indication of how much work they expect to introduce and buy when they believe they can afford to sustain this rate.

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.
- 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 a reasonable set of producer and consumer traits in real Grid settings.

2.2 Commodities Markets

In a real-world commodities market, a single type of commodity is exchanged in a central location. An important feature of the commodities market is that the goods brought to market by the various suppliers are regarded as interchangeable, market price is publicly agreed upon for the commodity regarded as a whole, and all buyers and sellers decide whether (and how much) to buy or sell at this price. Contrast this type of commerce with one based upon auctions, wherein each buyer and seller acts independently and contracts to buy or sell at a price agreed upon privately.

Since the goal of a computational Grid is to provide users with resources without regard to the particular supplier, it seems very natural to model a Grid economy using commodities markets. To do so, we require a pricing methodology that produces a system of price adjustments which bring about market equilibrium (i.e. equalizes supply and demand).

2.2.1 Dynamic Pricing in Commodities Markets

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. (For the exact formulation of the assumptions made in a market economy, see Debreu [13].)

A unique equilibrium price is guaranteed to exist in this framework by a theorem of Debreu ([13], Chapter 5), the proof of which is non-constructive and involves topological methods. Since Debreu's result, there have been various attempts to formulate rules for price adjustment which can be guaranteed to produce a sequence of prices which converges on the equilibrium price. Most notably, a variation of Walras' *tâtonnement* ("groping"; cf [31]), in which each price is adjusted individually in response to its own excess demand, can be shown to converge to economic equilibrium, but only under the restrictive hypothesis of "gross substitutability." By definition, a market economy exhibits gross substitutability if the demand for any good is non-decreasing in the price of any other good; that is, if an increase in price for good i, all other prices remaining constant, never causes the demand for good $j, j \neq i$, to decrease. Gross substitutability is not present in our market. For example, an increase in the price of CPU leads to a decrease in demand for CPU, which will also lead to a decrease in demand for disk, since CPU and disk are used in conjunction with one another. (A real-world example might be livestock and feed corn, since a higher price of cattle means fewer cattle purchased, which in turn lessens the need for feed corn.)

In [27], Smale produced a means for proving the existence of equilibrium which also entails a scheme for price adjustments which reaches it:

If commodity prices are represented as a *price vector* $\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix}$, 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 [33]. 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

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

2.2.2 Price Adjustment Schemes

Herein we examine the results of using several price adjustment schemes in simulated computational market economies. Smale's method is not possible to use directly for a number of reasons. First, any actual economy is inherently discrete, so the partial derivatives in equation 5 do not exist, strictly speaking. Second, given the behavior of the producers and consumers described above, there are threshold prices for each agent that bring about sudden radical changes in behavior, so that a reasonable model for excess demand functions would involve sizeable jump discontinuities. Finally, the assumptions in Smale's model are that supply and demand are functions of price only and independent of time, whereas in practice there are a number of ways for supply and demand to change over time for a given price vector.

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."

Implementing Smale's method: As observed above, 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. Starting with a price vector, we find their preferences at price vectors obtained by fixing all but one price and varying the remaining price slightly, thus achieving a "secant-line" approximation for each commodity separately; we then substitute these approximations for the values of the partial derivatives in the matrix $D_{\mathbf{z}}(\mathbf{p})$, discretize with respect to time, solve Equation 5 for the increment $d\mathbf{p}$ to get our new price vector, and iterate. We will refer, conveniently but somewhat inaccurately, to this price adjustment scheme as *Smale's method*.

The First Bank of G: The drawback to the above 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 the using *tâtonnement* (see above) 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 decried above. For all simulations described in Section 3, we use a degree 17 polynomial.

2.3 Auctions

Auctions have been extensively studied as resource allocation strategies for distributed computing systems. In a typical auction system (e.g. [11, 29, 21, 6]), resource producers (typically CPU producers) auction themselves using a centralized auctioneer and sealed-bid, second-price auctions. That is, consumers place one bid with the auctioneer, and in each auction, the consumer with the highest bid receives the resource at the price of the second-highest bidder. This is equivalent to "just" outbidding the second-highest bidder in an open, multi-round auction, and encourages consumers to bid what the resource is worth to them (see [6] for further description of auction variants).

When consumers simply desire one commodity, for example CPUs in Popcorn [21], auctions provide a convenient, straightforward mechanism for clearing the marketplace. However, the assumptions of a Grid Computing infrastructure pose a few difficulties to this model. First, 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. This is expenditure is wasteful, and the uncertain nature of auctions may lead to inefficiency for both producers and consumers.

Second, 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 2.1.1 above. Consumers use the demand function as described in Section 2.1.3 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 resource submissions for a time step, they cycle through all the commodities in a random order, performing one 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 a second-highest bidder whose price is greater than the commodity's minimum price, then the price for the transaction is the second-highest bidder's price. If there is no such 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. When a consumer and commodity have been matched, the commodity is removed from the auctioneer's list of commodities, as is the consumer's bid. At that point, the consumer can submit another bid to that or any other auction, if desired. This occurs when a consumer has obtained all commodities for its oldest uncommenced job, and has another job to run. Auctions are transacted in this manner for every commodity, and this process is repeated at every time step.

Note that this structuring of the auctions means that each consumer may 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 price 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 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 all ranges (e.g. slots per CPU), uniform pseudo-random numbers were drawn from between the given extrema. 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.

Using our simulated markets, we wish to investigate three questions with respect to commodities markets and auctions.

1. *Do the theoretical results from Smale's work [26] apply to plausible Grid simulations?*

2. *Can we approximate Smale's method with*

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

one that is practically implementable?

3. *Are auctions or commodities markets a better choice for Grid computational economies?*

Question (1) is important because 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 the answer to question (1) is affirmative, we also wish to explore methodologies that achieve or approximate Smale's results, but which are implementable in real Grid settings. Lastly, recent work in Grid economies [1, 16, 24] and much previous work in computational economic settings [12, 22, 5, 30] has centered on auctions as the appropriate market formulation. We wish to investigate question (3) to determine whether commodities markets are a better, or at least plausible alternative.

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 con-

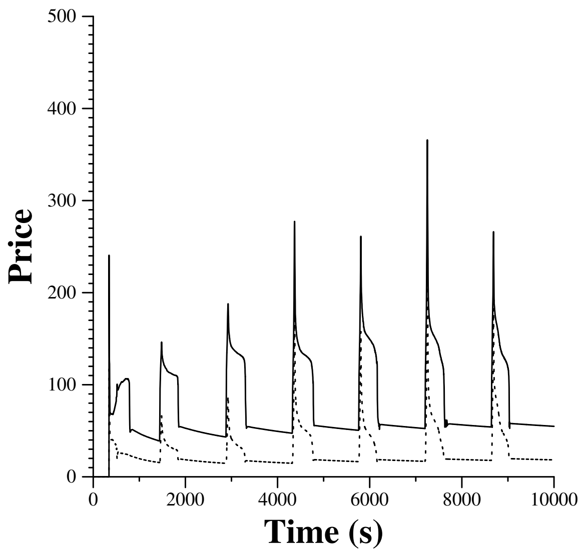

Figure 1. Smale's prices for the under-demand case. Solid line is CPU price,' and dotted line is disk price in \$G

sumer job submission is 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 excess demand spikes in conjunction with the diurnal load, but is quickly brought to zero by the pricing shown in Figure 1 where it hovers until the next cycle. Figure 3 shows excess demand for disk during the simulation period. 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 4 we show the pricing determined by our engineering approximation to Smale's method — the First Bank of G The First Bank of G pricing closely approximates the theoretically achievable results generated by Smale's method in our simulated environment. However, The Bank does not require polling to determine the

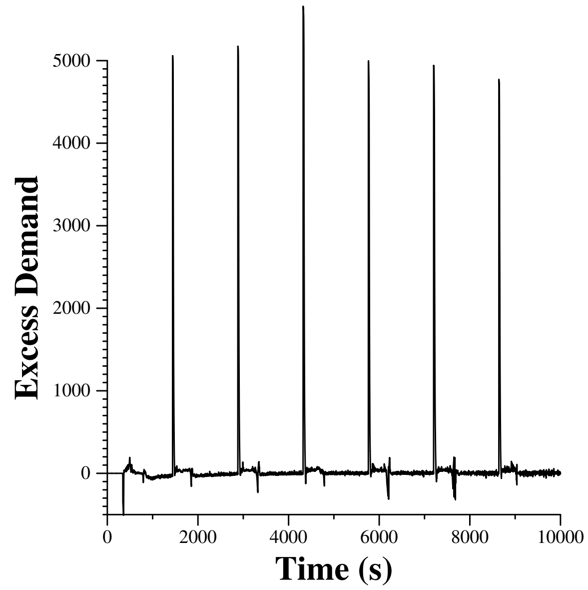

Figure 2. Smale's CPU excess demand for the under-demand case. The units are CPU slots.

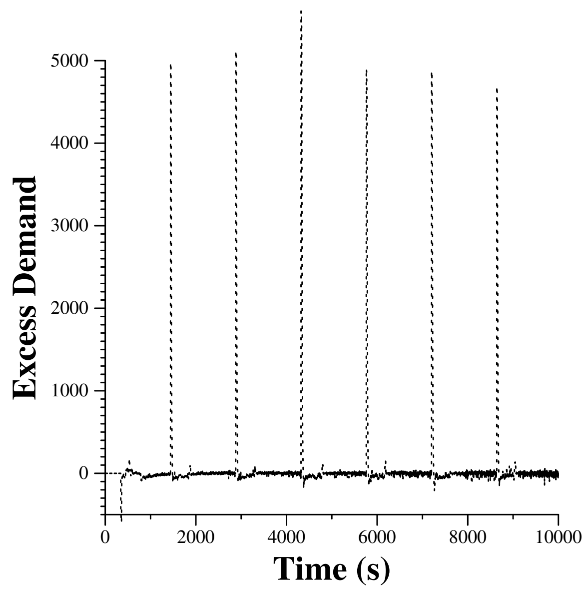

Figure 3. Smale's disk excess demand for the under-demand case. The units are simulated file units.

partial derivatives for the aggregate supply and demand functions. Instead, it uses an iterative polynomial approximation that it derives from simple observations of purchasing and consumption. As such, it is possible to implement the First Bank of G for use in a real Grid setting. Figures 5 and 6 show 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 Figures 7 and 8 (we show CPU and disk price separately as they are almost identical and obscure the graph when overlayed). 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 prices set by the commodities market. 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 is virtually identical. Disk resources are more plentiful in out simulations so disk prices should be lower in a healthy economy. The auction fails to capture this relationship, but the commodities market (both theoretically an 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. Under this assumption, the market is in equilibrium when all bids are satisfied (demand is satisfied) and all auctioned goods are sold (supply is exhausted). Any surplus goods or unsatisfied bids are "excess." While is does not make sense to assign a sign to these surpluses (surplus supply, for example, may not be undemanded supply) in the way that we can with aggregate supply and demand in a commodity market, in absolute value this mea-

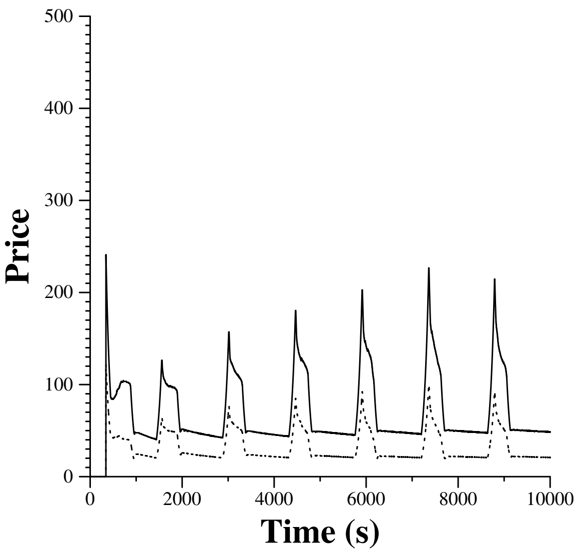

Figure 4. First Bank of G prices for the under-demand case. Solid line is CPU price,' and dotted line is disk price in $G

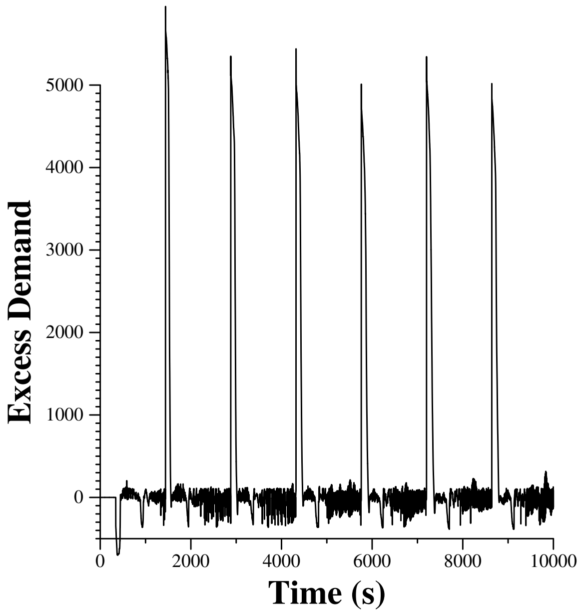

Figure 5. First Bank of G CPU excess demand for the under-demand case. The units are CPU slots.

sure captures distance from equilibrium. Hence we term it *absolute excess demand*.

In Figure 9 we show this measure of excess demand for CPUs in the under-demanded auction. Figure 10 shows the same data as in Figure 5 from the First Bank of G, but in absolute value. While the First bank of G shows more variance in absolute excess demand, it achieves equilibrium at times. Conversely, the auction sets prices that never satisfy the market. Strangely, the auction comes closest to equilibrium when demand spikes at each day-break. We are working to understand this behavior and will report on it as part of our future endeavors.

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 [30] 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 we similar prompting us to omit their bulk in favor of space, with one exception. Figure 11 shows the pricing information using Smale's method for the over-demand market, and Figure 12 shows the prices determined by the First Bank of G. Note that Smale's method determines a higher price for disk than CPU and that the First Bank of G (which correctly identifies the CPU as the more expensive commodity) chooses a significantly higher price for CPU, but a lower price for disk. While it

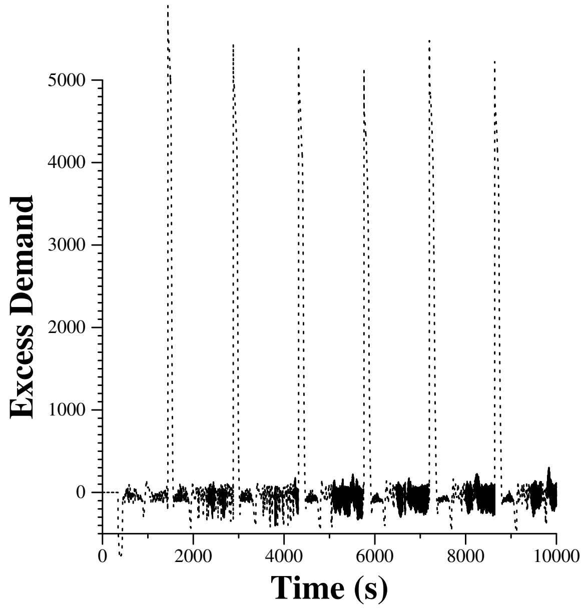

Figure 6. First Bank of G disk excess demand for the under-demand case. The units are simulated file units.

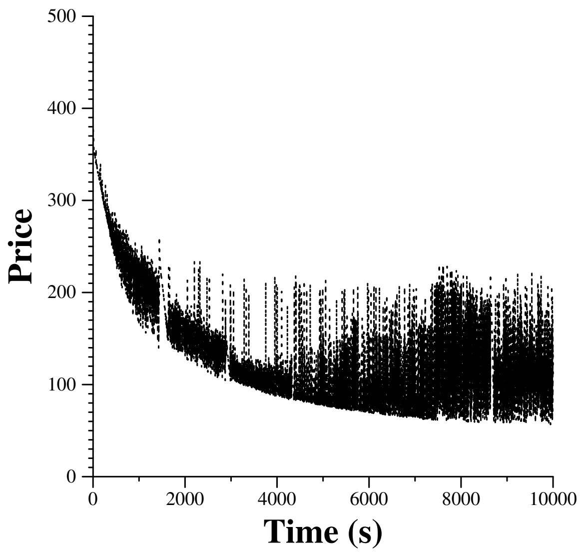

Figure 7. Auction prices for the under-demand case, average CPU price only, in $G

seems that one method or the other as determined a non-equilibrium price, excess demand graphs (Figures 13 and 14) for CPU show that they both are centered on market equilibrium. While it is difficult to read from the graphs (we use a uniform scale so that all graphs in this study of a certain type may be compared), the mean excess demand for the data shown in Figure 13 is 52.4, and the the First Bank of G data in Figure 14, the mean excess demand is 25.6. Both of these values are near enough to zero to show equilibrium, we believe, but the price inversion is difficult to explain. We conjecture that Smale's method, which relies upon probing the market to observe partial derivatives, becomes insensitive to price fluctuation near market saturation. That is, in the over-demand case, small price changes to do not yield meaningful changes in excess demand (the demand is constant and high) so partial derivatives may not be observed. The First Bank of G, however, uses tâtonnement when excess demand exceeds a pre-specified threshold causing it to drive the price in the "right" direction. The First Bank of G, then, will continue to raise the price until demand is extinguished. It is worth noting, however, that neither method centers excess demand far from zero and market equilibrium.

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. Since CPUs are the scarce resource, disk price may fluctuate through a small

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

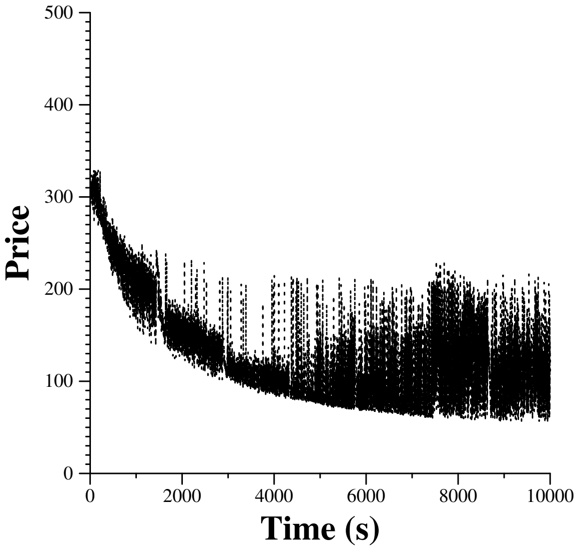

Figure 8. Auction prices for the under-demand case, average disk price only, in $G

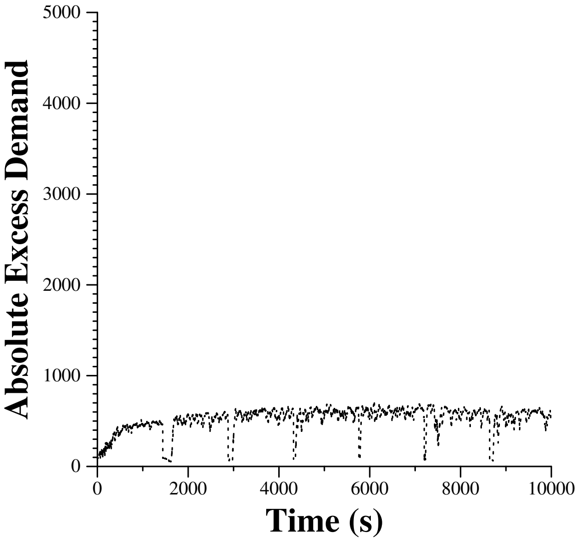

Figure 9. Auction absolute excess demand for CPU in the under-demand case. The units are CPU slots.

range without consequence when lack of CPU supply throttles the system. The auction seems to achieve slightly better disk utilization under these conditions. In general, however, Smale's method and the First Bank of G approximation both outperform the auction in the simulated Grid setting.

4 Conclusions and Future Work

In this paper, we investigate G-commerce — computational economies for controlling resource allocation 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). While there are an infinite number of ways to represent individual resource supply and demand in simulated setting, and none are completely accurate, we have identified a set of traits that we believe are realistic.

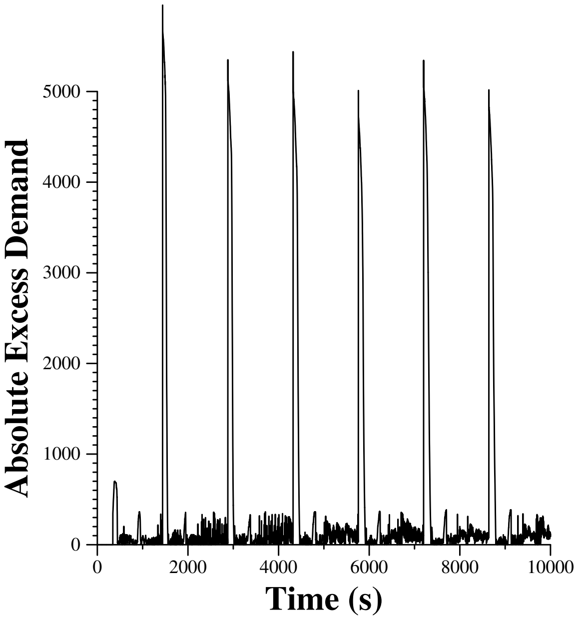

Figure 10. First Bank of G absolute excess demand for CPU in the under-demand case. The units are CPU slots.

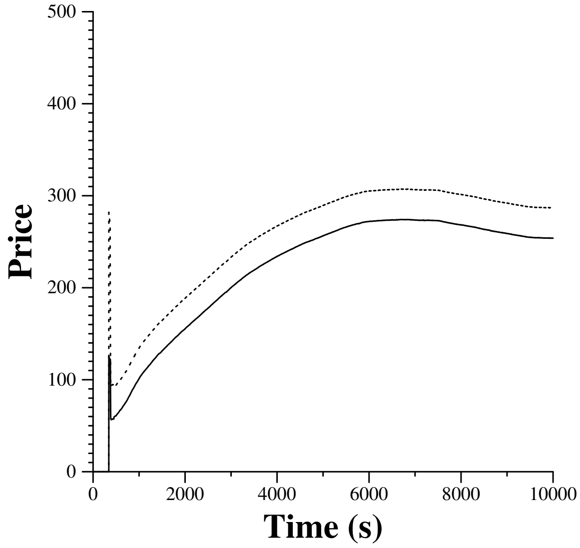

Figure 11. Smale's CPU and disk prices for the over-demand case. Solid line is CPU price, dotted line is disk price, and The units are $G.

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

Using simulated consumers and producers obeying these constraints, we investigate two market strategies for setting prices: commodities markets and auctions. Commodities markets are a natural choice given the fundamental tenets of the Grid [15]. 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,a nd consumer efficiency achieved by three methods in simulation. The first implements the theoretical work of Smale [26] which describes how to adjust prices in a commodities market to achieve equilibrium. It is viable in simulation, but impractical in the "real-world" as it relies on being able to poll reliably producers and consumers for supply and demand information. Often they do not know, or will not say what their response to a given price will be. The second method (The First Bank of G) is an implementable approximation to Smale's method. It uses a large-degree polynomial to approximate excess demand functions instead of polling making it parameterizable by observed market behavior only. Lastly, we simulate auctions in the style that has been investigated previously.

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.

As part of our future work, we plan two parallel thrusts. First, we are exploring the space of

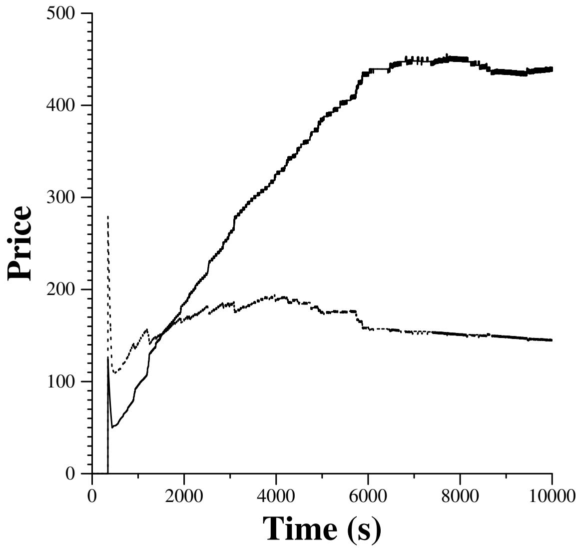

Figure 12. First Bank of G CPU and disk prices for the over-demand case. Solid line is CPU price, dotted line is disk price, and The units are $G.

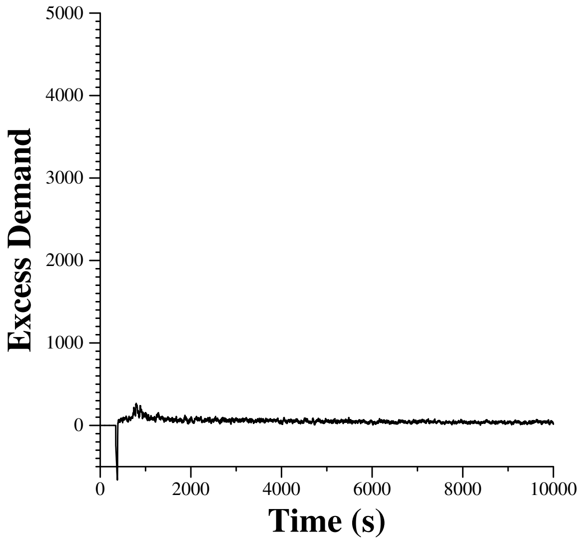

Figure 13. Smale's CPU excess demand for the over-demand case. The units are CPU slots.

plausible G-commerce formulations. Our goal is to identify and test, in simulation, different possible economies for the Grid. Secondly, we plan to construct a working version of the First Bank of G. Our previous work with the Network Weather Service [32, 34] and IBP [23] leaves us with the infrastructure necessary to build a large scale supply and demand information repository. Using the First Bank of G, we can generate prices based on "live" supply and demand information.

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. Technical Report PCS-TR97-326, Dartmouth College, Computer Science, Hanover, NH, Nov. 1997.

[6] 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.

[7] 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.

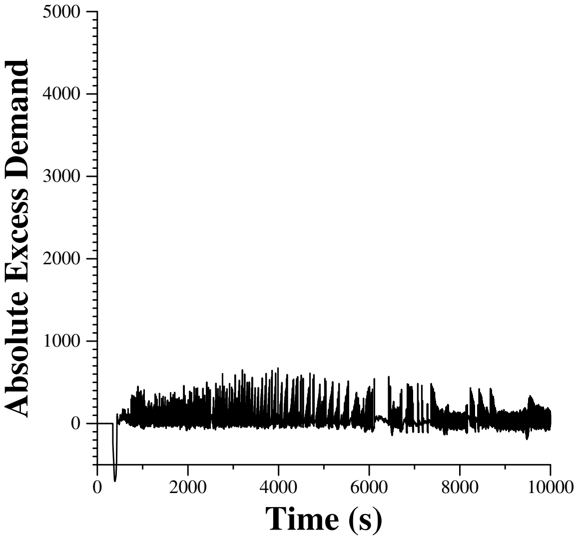

Figure 14. First Bank of G CPU excess demand for the over-demand case. The units are CPU slots.

[8] 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.

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

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

[11] 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.

[12] B. N. Chun and D. E. Culler. Market-based proportional resource sharing for clusters. Millenium Project Research Report, Sep. 1999.

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

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

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

[16] 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.

[17] 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.

[18] 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.

[19] J. M. M. Ferris, M. Mesnier. Neos and condor: Solving optimization problems over the internet. Technical Report ANL/MCS-P708-0398, Argonne National Laboratory, March 1998. `http://www-fp.mcs.anl.gov/otc/Guide/TechReports/index.html`.

[20] H. Nakada, H. Takagi, S. Matsuoka, U. Nagashima, M. Sato, and S. Sekiguchi. Utilizing the metaserver architecture in the ninf global computing system. In *High-Performance Computing and Networking '98, LNCS 1401*, pages 607–616, 1998.

[21] 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.

[22] N. N. Ori Regev. The popcorn market - an online market for computational resources. First International Conference On Information and Computation Economies. Charleston SC, 1998. To appear.

[23] J. Plank, M. Beck, and W. Elwasif. IBP: The internet backplane protocol. Technical Report UT-CS-99-426, University of Tennessee, 1999.

[24] B. Rajkumar. Ecogrid home page `http://www.csse.monash.edu.au/~rajkumar/ecogrid/index.html`.

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

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

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

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

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

[30] 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.

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

[32] R. Wolski. Dynamically forecasting network performance using the network weather service. *Cluster Computing*, 1998. also available from http://www.cs.utk.edu/ rich/publications/nws-tr.ps.gz.

[33] R. Wolski, J. Plank, and J. Brevik. g-commerce – building computational marketplaces for the computational grid. Technical Report UT-CS-00-439, University of Tennessee, April 2000.

[34] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance fore casting service for metacomputing. *Future Generation Computer Systems*, 15(5-6):757–768, October 1999. available from `http://www.cs.utk.edu/~rich/publications/nws-arch.ps.gz`.