g-Commerce – Building Computational Marketplaces for the Computational Grid

Rich Wolski† James S. Plank† John Brevik‡

† Department of Computer Science
University of Tennessee

‡ Mathematics and Computer Science Department
College of the Holy Cross

April, 2000

University of Tennessee Technical Report UT-CS-00-439
http://www.cs.utk.edu/~plank/plank/papers/CS-00-439.html

For the next piece of work on G-Commerce, including detailed simulations of pricing schemes based on commodities markets and auctions, see Tennessee Technical Report UT-CS-00-450 This technical report may be obtained on the web at: `http://www.cs.utk.edu/~plank/plank/papers/CS-00-450.html`

1 Introduction

With the proliferation of the Internet comes the possibility of aggregating vast collections of computers into large-scale computational platforms. Research driven by this realization has yielded a new software architecture, known as *The Computational Grid* [16], for building high-performance distributed applications and systems. The vision outlined by its architects is for applications to draw computational "power" from a distributed pool of resources in an analogous way to the way in which household appliances draw electrical power from a power utility: seamlessly and ubiquitously. More recent proof-of-concept work has demonstrated that the Grid paradigm is, indeed, a viable model for high-performance, wide-scale distributed computing [14, 36], but these initial efforts have exposed many significant research challenges.

In particular, the problem of allocating and managing system resources in a Computational Grid setting is a critical impediment to the development of a generalized Grid execution environment. Current Grid software infrastructures [7, 15, 19, 22, 24, 33] do not provide distributed and robust resource allocation and accounting mechanisms. Grid application users must either appeal to a centralized resource allocation broker [11, 17, 4] or they simply contend wantonly amongst themselves for shared resources. Centralized allocation systems will not scale in proportion to the potential computational power that will become available as high-performance networking becomes pervasive. Uncontrolled resource acquisition and release by individual users, conversely, can and will lead to large performance oscillations and potential system-wide failure, much as uncontrolled TCP/IP implementations led to congestion-induced network collapse during the mid-1980s [21]. Moreover, the problem of maintaining Grid stability in the presence of resource contention will be exacerbated by the deployment of automatic program schedulers such as AppLeS [5] and Autopilot [29]. *At present, there are no scalable resource allocation and control mechanisms or policies that can ensure both high-performance and stability in Computational Grid settings.* We believe that without them true wide-scale distributed computing will not be realized.

Proposal Thesis:

We propose to study, develop, implement and deploy economically based systems for allocating resources on the Computational Grid.

We term this research *g-Commerce* since it relies on economic principles — the relationship among supply, demand, and price — and the concept of resource purchase to control the allocation of Computational Grid resources.[1] Our work will investigate economic policies based both on fixed and dynamically varying pricing strategies for resources. The goal is to define a policy or set of policies that can be implemented efficiently without a centralized allocation broker and that will ensure the stability of the overall Grid execution environment.

This work will be both theoretical and experimental. The theoretical component will consist of developing and evaluating economic pricing models, relating them to the problem of Grid resource allocation, and proving properties such as stability, optimal balance of supply versus demand, and robustness in the presence of inconsistent information, incomplete information, and resource failures.

The experimental component will be to build *The First Bank of G.* This system will be a computational "exchange" that supports scalable resource allocation in dynamically varying distributed environments, and it will be modularized so that it can implement a variety of economic policies. We will use The First Bank of G to study those policies having the most attractive theoretical properties in real-world Grid settings. In particular, we will use the University of Tennessee's Scalabale Intracampus Research Grid (SInRG) [12] (recently funded by NSF) as a campus-wide Grid testbed. This facility will allow us to test the effectiveness of our policies, refine them according to the requirements of Grid software and resources, and perhaps influence the resource allocation, monitoring and scheduling components of different Grid software components. We will later make the fruits of this project available to a larger community through the National Partnership for Advanced Computational Infrastructure (NPACI), within which the PI is a partner. As such, The First Bank of G will serve as a development vehicle for policy and scheduling research, and as a persistent software artifact that we will make available to the Grid research community.

We believe that the unique application of economic policies to decentralized resource allocation is a valuable and necessary approach to make the Grid a viable architecture for large-scale computing using shared and federated resources. The following sections describe our research plan in greater detail. In the next section (Section 2) we describe resource allocation and some of the problems it poses in Computational Grid settings. Section 3 details the technical approach we plan to pursue, and Section 4 describes the ways in which we plan to test, verify and deploy our results.

2 Problem Statement: The Need for g-Commerce

Effective resource allocation strategies are difficult to design and implement for Grid environments. In this section, we briefly describe grid computing environments, and then motivate the need for our work.

2.1 Grid Computing

The Computational Grid [16] is a software architecture for supporting high-performance distributed computing using resources that are culled from multiple autonomous sites. Resources (machines, network connections, storage, etc.) are committed to a virtual, shared pool by their owners. Applications (under the control of Grid users) draw the resources they need from the resource pool automatically. The vision is analogous to the way electrical devices draw electricity from a power utility. Power producers (resource owners in the Computational Grid model) link their production facilities to an electrical grid (Computational Grid) managed by a power utility. Power consumers (applications in the Computational Grid setting) interface to the grid via a well defined interface and draw power without regard for how it was produced.

Since resources are *federated* from different ownership domains, the architecture does not assume that a single, authoritarian scheduling system will be implementable. Owners must maintain ultimate control over their resources. It is not feasible for the Grid software system to insist that those resources be subjugated to a centralized scheduling

[1] The "g" in "g-Commerce" is intended to signify that we are focusing on solutions for the Grid in the same way that the "e" in "e-commerce" indicates an Internet focus.

authority that is responsible for allocating resources to applications. Rather, resource owners and application users are expected to act independently, and largely in their own self-interests.

An important assumption made by the architects of the Computational Grid is that it will not be possible to force resource providers to remove or replace elements of their existing software base. Resource owners are often reluctant to replace their local system software for fear of destabilizing working systems, even if the software offers greater potential functionality. Therefore, to gain acceptance, the Grid must accessible as a set of services layered *on top of* installed, extant software rather than as a replacement. This software veneer is often termed *middleware* to indicate its position between the application and the local system software.

Several software infrastructures are currently being developed to implement the Grid architecture [7, 15, 24, 19, 33]. Critical to their success is the availability of a resource allocation and control methodology that is stable, high-performance, available as middleware, and does not require a single over-arching authoritative system.

2.2 Resource Allocation and Accounting: Challenges

When designing a Grid resource allocation scheme, there are many challenges that must be solved. We present these challenges here, and then state our approach for solving them.

Problems with centralized approaches. Today, most Grid middleware includes the notion of a centralized resource "broker" that is responsible for fielding user or application requests for resources and making resource assignments [7, 15, 19, 33]. Centralized resource brokers are convenient because they have one consistent view of their domain, and thus can make informed resource allocation decisions quickly.

However, there are many inherent problems with centralized resource brokers. First, they can be a performance bottleneck if many resource requests are made simultaneously. As a result, centralized approaches, while initially useful as prototypes, do not scale. Further, if a user requires resources from multiple administrative domains, it may be impossible for those resources to be granted from a single, over-arching resource broker. And finally, the centralized broker constitutes a single point of failure for the system. For a resource allocation to meet the needs of the Computational Grid it must be decentralized and distributed.

Problems with decentralized approaches. Decentralized approaches are difficult to manage, leading to potential system instability as resources are added to and removed from the Grid dynamically and as performance fluctuates due to contention. Inconsistent information about resource availability may prompt bad resource allocation decisions that are later retracted. If a retraction and subsequent reallocation of resources causes further retractions and reallocations, the system will oscillate.

Low overhead. The speed with which resources can be allocated and deallocated is also an important consideration. Any delay an application incurs while it waits for a resource assignment is perceived by the Grid user as overhead. Therefore, there is a tradeoff between the benefit of using the best resources that are available (from a bottleneck centralized broker, perhaps) and the benefit of using a suboptimal resource that can be located and allocated quickly (using a loosely-consistent distributed allocation service).

Heterogeneity. Resource heterogeneity introduces additional complexity. At an abstract level, a Grid application has a set of homogeneous resource requirements. That is, an application potentially requires:

1. a set of processing instructions to be executed;
2. a set of program data to be moved between application components to facilitate that execution; and
3. a set of data items to be read from and written to persistent storage.

It is the purview of the Grid software (and not the application's user) to meet these requirements using Grid resources. The Grid must choose the resources that best match the application's needs transparently, in the same way that a true power grid routes electricity from the best producer to each consumer's home. That is, a household appliance does not "know" how the power it is consuming was generated (i.e. via hydro-electric, fossil fuel, geothermic, wind, etc.), nor is it rendered non-functional if only a particular kind of power is available.

To make an assessment of how to meet an application's resource needs, given a pool of heterogeneous and shared resources, the Grid software must be able to assign a relative value to each resource. For example, if an application is particularly well suited to execution on a workstation (e.g. it is a sequential algorithm with small to moderate memory requirements and limited storage requirements) it will also likely run well on a single processor of a parallel machine such as an IBM SP2. If the processing is primarily floating-point, it may also be able to achieve a high absolute speed on an expensive vector processor such as a Cray T90. It is up to the Grid software to decide which of these resources should best be assigned to execute the application.

Resource Cost vs. Performance. As part of the Application-Level Scheduler (AppLeS) project [2, 5, 32], we have constructed scheduling agents that make allocation decisions based strictly on deliverable performance. The AppLeS scheduling agent evaluates each resource that is available by considering how quickly it can execute the application (measured in wall-clock time). That is, the agent uses the inverse of the turn-around time as a value metric, and chooses the most valuable resource. If the Cray T90 is the fastest in absolute terms, it is chosen even if the the code will use a small fraction of the T90s total absolute capability. While this scheduling approach has proved to be effective [32], it relies on the assumption that *all resources available to the application may be used with the same cost to the application's user*. The AppLeS agent only considers the benefit each resource provides to its application – since the cost is fixed and uniform, it is not considered.

If the cost associated with resource use is not uniform, however, the value of a resource to an application can no longer be determined strictly in terms of its benefit. The question of a resource's cost and value is then inherently an economic one, and motivates us to turn to solutions based in economics.

Non-independence of resources. Most applications need more than one resource from the Grid. As described above, each application has needs in processing, storage and communication, and the performance of each of these entities is dependent on the other. For example, a fast CPU in California may work better with a slow disk in California than a fast disk in Tennessee. The resource broker and application scheduler must work in tandem to ensure that the coupling of resources is considered in the resource allocation.

Stability. Particular care must be taken when resources are allocated to ensure overall system stability. Errors may result in either an overestimation of the resource value or an overestimation of its cost. If an application (under the control of a user or an automatic scheduler) corrects its resource allocation when it detects these errors, it may waste time and resources oscillating between resources that appear to have the lowest cost-to-benefit ratio at any given time.

Accounting. The lack of credible cross-domain resource accounting for Grid environments poses a serious impediment their success. The Grid model of resource aggregation is a federated one. Resource owners commit the usage of their resources to the Grid but retain ultimate control and revocation rights. Without an effective computational economy, however, it is not possible for resource owners to charge or account for the the resources they have committed to the Grid. For example, consider the possibility of combining resources from the National Partnership for Computational Infrastructure (NPACI) and the National Computational Science Alliance (NCSA) – two infrastructure partnerships maintained by the National Science Foundation. Currently, it is not possible for a user with an allocation of IBM SP2 time at NPACI to "trade" some of that time for an equivalent amount of Origin 2000 time at NCSA even if the Origin 2000 would execute the user's job substantially more efficiently (both in terms of execution performance and allocation cost).

2.3 A Solution Based on Economic Pricing Models

The reason the above trade is not possible it is that it is not possible to determine the relative worth of time on the SP2 to time on the Origin 2000. If dynamic pricing information for both the SP2 and the Origin 2000 were available, and the prices reflected the true value to the Grid of each, it would be possible for the NPACI user to purchase some Origin 2000 time at a cost to his or her NPACI allocation.

Clearly, a computational economy for the Grid is a natural way to address many of the challenges posed by its dynamically changing, heterogeneous, and federated structure. Distributed economic models can be evaluated efficiently ensuring low resource allocation overhead. They allow resource producers and consumers to act in favor of their self-interest with a minimal of prior agreement. Formal theories of pricing and practical experience with real-world markets yield a variety of pricing strategies that are likely to ensure stability ([3], [31]). With accurate resource pricing comes the ability to realize credible accounting systems between federated resource owners.

For these reasons, we propose to study computational economies as a basis for a decentralized, distributed resource allocation mechanism for the Grid. Note that other research efforts [1, 6, 34, 26] have considered the use of auctions as a methodology for controlling resources in agent-based systems. While these approaches may offer benefits with respect to user-proxy agents, they do not address the performance or stability requirements that are essential to making the Grid a success.

3 Technical Approach

The goal of our technical approach is to develop both a solid theoretical understanding of Computational Grid resource allocation strategies that are economically based, and to provide an implementation of a working "computational exchange" that will enable further computational economics research. Our plan is to identify promising economic formulations for Computational Grid settings, and to study their stability characteristics and implementational feasibility in for the Grid. We will then verify those models that prove most attractive using simulation. At the same time, we will develop a software infrastructure for the Grid that can be parameterized by different economic policies. This *First Bank of G* will permit us to study different economic systems in "live" Grid settings and will serve as a useful development environment for future Grid economic research. While a research plan that rests both on theoretical grounds and on verification through implementation is ambitious, our research team will call upon expertise in mathematical systems (Brevik) and Grid development (Wolski, Plank) to achieve success.

3.1 Theory: Defining A Computational Economy with Dynamic Pricing

To model resource usage in a Computational Grid setting according to a set of economic principles, we must

- determine which computational resources will be treated as commodities or traded goods,
- specify units of supply and demand for each commodity, and
- devise a system for setting prices that is based on the relationship between supply and demand.

Price changes cause changes in supply and demand levels, which may, in turn, cause price changes. If a stable price for all commodities cannot be reached, the system will oscillate due to continuous adjustments to the price and supply/demand levels. More seriously, if resource allocation decisions are based on these levels, it will not be possible to determine a stable resource allocation for a given set of Grid resources. We must, therefore, design computational economies, based on pricing systems and supply/demand measurements, that we can show will reach equilibrium. This outlook differs from other approaches to computational economies (Millennium [1], Spawn [34], D'Agents [6], Popcorn [26]), which have favored auctions as a means of price-setting and are not concerned with price stability or equilibrium. Indeed, in [6], the idea of market-based price adjustments is described as "simple, ad hoc, and information-poor." On the other hand, in [34], it is demonstrated through trials that auctions lead to substantial fluctuation in selling prices, and this is recognized as a considerable problem from the standpoint of producing a stable, functioning economy.

Of special interest is work by Chun and Culler [9], where CPU timeshare allocations are governed by a market economy that optimizes *user value*. Their approach does not address the issue of market equilibrium; rather, as with auctions, it sets prices locally, but prices are set based on priority. Although the resource model and assumptions in this work differ from ours, we share the same basic philosophy (that a market economy is best for distributed resource allocation), and will try to leverage the relevant results that emerge from this research.

3.1.1 The Basic Approach

We can model our problem mathematically by regarding the prices of the commodities (e.g. CPU's, storage, etc.) as entries in 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. For our purposes, each price p_i is a function of time. For a given price vector, the supply and demand calculated for each individual commodity, based on each producer's and consumer's evaluation of price, may or may not balance. Define the *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. Assuming that the markets for these commodities are interrelated, each z_j is a function of all of the prices p_i, that is, of the vector $\mathbf{p}$. Excess demand can again be written as a vector $\mathbf{z} = \mathbf{z}(\mathbf{p}) = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix}$. Now, *economic equilibrium* occurs when $\mathbf{z} = \mathbf{0}$. Thus, bringing a particular market to equilibrium is equivalent to finding a price vector for which each excess demand is 0.

The method employed here is a generalization of Newton's method. With this method, one iteratively constructs approximate solutions to a system of equations by solving the linearizations of these equations at one approximate solution to obtain the next approximate solution.

According to a theorem of Smale [31], given a market consisting of n interrelated commodities with price vector $\mathbf{p}$ and associated excess demand vector $\mathbf{z}(\mathbf{p}) = \mathbf{z} =$ as above, an *equilibrium point* (that is, $\mathbf{p}^*$ such that $\mathbf{z}(\mathbf{p}^*) = 0$) exists; further, under a mild set of boundary hypotheses which, in our application, we are able to control, this equilibrium is reached from any starting price vector as follows: For any value of $\mathbf{p}$, 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 vector equation

$$D_{\mathbf{z}}(\mathbf{p})\frac{d\mathbf{p}}{dt} = -\lambda\mathbf{z}(\mathbf{p}). \tag{1}$$

Since the partial derivatives amount to local linearizations, Smale's method can be thought of as a "continuous" Newton's method, where prices are being updated continuously. Since our problem involves discrete price updating, our algorithm will be based on the Euler discretization of the differential equation above. In our discretized application, with all the d's replaced by Δs, λ will have units of inverse time and its value will be chosen to cancel Δt, so in practice λ will be taken to be 1 and the factors of Δt omitted. (This is just to say that our price update will be independent of how much time has elapsed since the last update, which is necessary within our framework since there is no time derivative within the matrix $D_{\mathbf{z}}(\mathbf{p})$ on which our calculations are based. See the example below.) Newton's method is particularly attractive in a Grid setting as many high-performance, distributed implementations have been developed.

Applying this to grid resource brokering, we partition our resource space into collections of commodities, where each collection corresponds to an entry i in $\mathbf{p}$ and $\mathbf{z}$. A very coarse example would be to have three commodities: CPU's, storage, and network bandwidth. A more realistic example would be to have a finer partitioning made, for example, according to resource performance and network location. For each commodity collection, we must have a function that determines $\mathbf{z}$ from $\mathbf{p}$. This function may be approximate and calculated, for example, from monitoring resource availability, demand, and usage over time, and applying statistical models to the data [35, 39]. Alternatively, it may be provided by resource producers and consumers as they specify what they are willing to charge or pay (respectively) for resources. Using this function and the above algorithm, we calculate a new $\mathbf{p}$ such that $\mathbf{z}$ is approximately zero, and that determines the price for the resource collections. Users (and scheduling agents) may then use these prices to allocate resources.

Smale's approach is attractive from an algorithmic standpoint, because the price adjustments are based on local behavior of the agents (buyers and sellers). This local quality is important for two reasons. First, it allows for simpler calculations than does a more standard model of "Walrasian" equilibrium, which involves long-term optimization of utility value for the agents (see [31]). Second, it relies only on local knowledge about supply and demand functions rather than perfect knowledge on a global level of these functions. From a theoretical standpoint, it is at the same time concise and general enough to guarantee convergence in a very general framework.

In sum, from an economic standpoint, our goal is to achieve convergence to market equilibrium. (If we were only concerned with the more modest goal of clearing the market at each available opportunity, then it would be appropriate to use auctions to set price.) By structuring our g-Commerce economy so that the participants agree to buy and sell at fair market value, we achieve a computational economy that is fair, robust, *and stable*.

3.1.2 An Example

Suppose that in our market there are three commodities, say (1) CPU, (2) live memory, and (3) disk, and that the current market prices, in "Grid dollars" (G$), are G$3, G$4, and G$5 respectively. Suppose further that the current levels of excess demand are, again respectively, 1000, -2000, and 0, and that each price increase of G$1 in each commodity has been measured, at prices near this level, to reduce the demand for this commodity by 500 and increase the supply by 500, giving a decrease of 1000 in excess demand. Finally, suppose that each increase of G$1 in each commodity results in *a decrease of 50 in the demand of each of the other two commodities,* resulting from budgetary constraints of the users and also the fact that if, for instance, a user plans to use less CPU, then that user's need for memory might decrease as well. Note that this constraint reflects the above claim that the excess demand function of each commodity is in fact a function of the entire price vector and not just the price associated to that commodity. We can calculate our updated price as follows: Form the matrix

$$D_{\mathbf{z}}(\mathbf{p}) = \begin{pmatrix} -1000 & -50 & -50 \\ -50 & -1000 & -50 \\ -50 & -50 & -1000 \end{pmatrix}$$

and solve the discretized equation coming from Equation 1 above, which is

$$\begin{pmatrix} -1000 & -50 & -50 \\ -50 & -1000 & -50 \\ -50 & -50 & -1000 \end{pmatrix} \begin{pmatrix} \Delta p_1 \\ \Delta p_2 \\ \Delta p_3 \end{pmatrix} = \begin{pmatrix} 1000 \\ -2000 \\ 0 \end{pmatrix}$$

to give us the approximate values

$$\Delta p_1 = -.02, \Delta p_2 = 1.02, \Delta p_3 = -.55,$$

so our new pricing will be p_1 =G$2.98, p_2=G$5.02, p_3 =G$4.45.

Note that our new prices are approximations of equilibrium prices based on linearizations of the various market sensitivities, which we certainly cannot assume to be linear. Thus we say that our procedure *converges* to equilibrium prices rather than finding them immediately and precisely.

■

3.1.3 Extending Further

While Smale's result gives us a useful starting point for building a robust, stable computational economy, it is based on the assumption that all quantities are continuous, and that perfect data for the partial derivatives at any given moment. What we are seeking is a *discrete* process of price updates based on *approximations* for the partial derivatives based on previous data points. A further issue is that at different locations the most recent available price data may not match. We wish to find a set of criteria which will guarantee the existence of a convergent process and which we will be able to control in a distributed environment. In particular, we need to develop a convergent methodology that does not rely on a centralized "omniscient" process. To be successful, our pricing strategy must converge (eventually) when it is implemented by a set of distributed entities, each with an imperfect estimate of the global supply and demand state.

As part of this work, we will develop computational economic formulations that are provably stable in the presence of temporarily imperfect data. We believe that such formulations are possible based on an initial analysis of Smale's result although we plan to explore alternative strategies as well. For example, it is possible to model available network performance not as a commodity, but as an *asset* that affects the value of other commodities. For example, a CPU may become more or less valuable based on some measure of the network performance that links it to the Grid, but the network itself is not treated as a commodity. Generally, our approach to price adjustment will be quite flexible both in terms of *how* commodities are valued and also *what* the commodities actually are, so we are free to explore models which set network as a commodity as well as those which apply "value enhancement" to high-network commodities. By the same token, "packages" in various combinations (for example, a cluster of CPU's connected by a high-performance network) could also be brought to market as commodities in themselves, distinct from but related to all the other commodities in the market. A further direction for exploration is to incorporate the ideas about priority-based pricing in Chun and Culler [9] into the framework of market equilibrium.

In sum, we plan to study different economic formulations, both in terms of their dynamic properties and their suitability for implementation in a Grid setting.

3.2 Implementation: Building a Computational Exchange

To provide a system for setting prices dynamically and advertizing those prices to resource allocators and consumers, we plan to build a computational exchange. This *First Bank of G* will:

- Dynamically monitor resource supply and demand.
- Set prices according to an economic model that ensures stability and can be evaluated efficiently.
- Issue verifiable, time-stamped pricing information certificates.

Resource and application schedulers will thus be able to use pricing certificates to make dynamic scheduling decisions. An application user or automatic scheduler will be able to choose the cheapest set of resources that meet its performance needs. We will also explore the possibility of interfacing the First Bank of G with local accounting and allocation mechanisms as a way of linking resource supply and demand to actual user allocations.

Figure 1 depicts, in cartoon form, the structure of the system that we envision. Applications and resources from separate administrative domains (represented by the circles and squares in each corner of the the figure) are instrumented so that aggregate demand and supply can be recorded. We will store time-stamped supply and demand measurements in a distributed repository (cylinders in the figure) that can be accessed by a cooperating set of Branch Offices (shaded circles). Branch Offices exchange supply and demand information, set prices, and publish price data via a well-defined set of abstractions. We will implement a robust information exchange protocol that will permit Branch Offices (and other system components) to synchronize price, supply, and demand data providing a consistent view to all Bank of G clients.

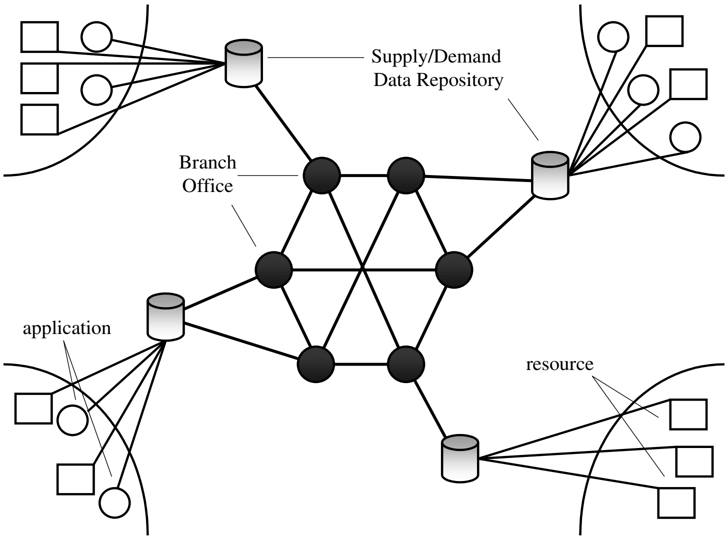

Figure 1: **First Bank of G Structure and Components**

3.2.1 Leveraging Previous Results

Our plan is to take advantage of other successful Grid tools and research results in building the First Bank of G. In particular, we will use the following building blocks as a basis for our implementation:

- The Network Weather Service (NWS) [35, 38] to serve as a repository for dynamic supply and demand information.

- The EveryWare gossip protocol [36, 37] to maintain loosely synchronized and replicated data.
- The Internet Backplane Protocol (IBP) [28] to manage time-limited network buffer storage.
- The Globus authentication protocol [18] to issue time-stamped, digitally signed *price certificates*.

Each building block will be described below by the functionality that it implements.

3.2.2 Gauging Resource Supply and Demand

We will modify the Network Weather Service (NWS) [35, 38] to serve as a data repository for dynamic supply and demand information. The NWS is a robust, distributed system for gathering dynamic resource and application performance information and applying fast statistical forecasting models to the data. Both measurement data and statistical forecasts are made available to automatic schedulers in near-real time. The NWS

- operates a distributed set of performance sensors, from which it periodically collects performance measurements,
- applies a set of statistical forecasting techniques to individual performance histories, and
- generates forecast reports for the resources being monitored, which it disseminates via a number of different APIs.

The goal of the system is to apply forecasting methodologies to "fresh" performance monitoring data gathered from Grid resources to make forecasts of available performance in near-real time. The current implementation of the NWS is fully distributed, robust, and compatible with a variety of Grid infrastructures [38] including Globus [15], Legion [19], NetSolve [7], Condor [33], and NINF [24] as well as MPICH [20], and native Unix sockets.

We will use the NWS data management subsystem as the basis for gathering, managing, and distributing instantaneous supply and demand readings (represented by the cylinders in Figure 1). To determine resource supply and demand at any given time, we will distribute a set of *supply sensors* among the available Grid resources, and a *demand sensor* library that can be loaded with Grid applications. Supply sensors are analogous to NWS resource performance sensors − they capture the quantity of resource that is available at any given moment either by probing the resource, or by reading existing dynamic performance data (e.g. Unix load average). We will also develop a set of demand sensors that capture an application's resource needs in terms of the resources that are available. Almost all Grid middleware systems such as Legion [19], Condor [33], NetSolve [7], Globus [15], etc. export a resource specification interface to the user or application scheduler. Our library will be compatible, initially, with the Globus Resource Specification Language (RSL) [10] as it is an emerging standard. By parsing the RSL, the demand sensor will be able to deduce the resource needs of the application, and to send those needs to the First Bank of G. Both supply and demand information will be time-stamped and stored in a distributed data repository using the existing NWS data management mechanisms [38]. We will modify the current NWS APIs to allow supply and demand information to accessed efficiently.

3.2.3 Calculating and Distributing Price Information

To calculate and distribute pricing information, we will develop a system of *Branch Offices* that query the supply and demand information base, calculate pricing information, and publish that information for Bank of G customers. Branch Offices will be able to calculate pricing information dynamically based on the economic models we find most promising. They will query the supply and demand data repository to gauge current conditions, possibly synchronizing with other Branch Offices, and they will circulate pricing information for publication (represented by the shaded circles in Figure 1).

It is critical that this system be distributed and robust with respect to network and host failure in order for it to be useful on a Computational Grid. At the same time, the pricing information must be consistent across Branch Offices so that all customers "see" the same prices. We will leverage the EveryWare *Gossip protocol* [36] for synchronizing replicated data structures in Grid settings (represented by the interconnected hexagon in Figure 1). EveryWare state-exchange servers (called *Gossip*s) allow application processes to register for state synchronization. The application component must register a contact address, a unique message type, and a function that allows a Gossip to compare the "freshness" of two different messages having the same type. All application components wishing to use the Gossip service must also export a state-update method for each message type they wish to synchronize. The Gossip servers tell each application that has outdated data where to get the freshest copy. Based on the variance in network performance between Gossips (available from the Network Weather Service) we can bound the synchronization time.

We will augment this research in two significant ways. First, we will improve the Gossip protocol using the Internet Backplane Protocol (IBP) [28]. IBP is a software tool for managing and making use of privately owned storage as a generic network resource. In the context of this research, IBP is a mechanism for allocating and managing time-limited and failure-prone network storage buffers. IBP has several design features that make it ideal for this purpose:

- **It uses capability-based naming**. With IBP, a client allocates a storage buffer, and receives a capability, or name, in return. This name is then presented to the IBP server for each storage or management transaction. As such, there may be no user-defined names in IBP. The elimination of user-defined names facilitates scalability, since no global namespace needs to be maintained. Moreover, capabilities are not recycled, so when a buffer is deleted, its capability becomes invalidated by default, again facilitating scalability.

- **Storage may be constrained to be *volatile* or *time-limited*.** An important issue when serving storage to Internet applications is being able to reclaim the storage. IBP servers may be configured so that the storage allocated to IBP clients is *volatile*, meaning it can go away at any time, or *time-limited*, meaning that it goes away after a specified time period.

- **Failures are tolerated gracefully.** Since failures are a fact of life in network applications, IBP has been designed to help users tolerate failures gracefully. In particular, IBP has been designed to be robust to client failures, network failures, and even processor failures at the storage servers.

- **Data can be placed in proximity to other resources.** Since IBP allows clients to allocate network storage buffers remotely, the clients can advantageously store or collect data near where the data is being either generated or consumed [28, 13]. This in turn can improve performance over systems where the explicit management of network storage is not performed. We call this form of networking, where network buffers are explicitly managed *logistical networking*.

By implementing the Gossip protocol over IBP, we allow our Branch Offices to gain all the advantages of IBP's design for use in networking applications: making use of storage with distributed ownership, employing time-limited storage allocation, tolerating failures gracefully, and employing logistical networking for improved performance.

The second augmentation to the Gossip protocol will be to study the effect of bounding the time required for a set of processes to synchronize replicants using our system. These time bounds may be required in order for our economic pricing model to achieve stability. The effect of the time bounds on stability of the system and on overall resource utilization will be something that we will attempt to quantify, either theoretically, experimentally, or both.

3.2.4 Issuing Price Certificates

Pricing information (generated by Branch Offices) will be available via published price certificates. Initially, we will focus on publishing the prices via C-language and Java APIs, and via a graphical, web-based interface. The APIs will be useful to automatic schedulers such as AppLeS [5, 32] and Autopilot [29], and the graphical interface will allow users to "see" the computational market. Experience with the NWS leads us to believe that a graphical interface will also serve as a useful debugging tool.

If application schedulers and application users agreed to use the cheapest resources whenever possible, resource utilization will be maximized since resource price is a function of supply and demand. However, if resource prices can be tied effectively to site accounting and allocation mechanisms, users and schedulers can make their allocation decisions based on an actual resource cost. For example, a user with access to machines located at both NPACI and NCSA resources has an allocation of local resource units assigned for each machine. If NPACI and NCSA agreed to honor the resource price determined (at any given time) by the First Bank of G for each of its machines, a user could choose the machine(s) that reduced his or her combined allocation by the least amount. Both NPACI and NCSA would need to be able to convert a First Bank of G price into some number of resource allocation units so that the user is effectively charged for his or her usage. That is, a user with SP2 time at NPACI could choose the Origin 2000 at NCSA (because it was cheaper at the time) and have his or her SP2 allocation decremented by the current price.

To make cross-domain accounting possible, price certificates must be reliably time-stamped and verifiable so that local accounting systems can trust them. We plan to use standard clock-synchronization protocols such as NTP [23] to provide a Grid-wide time stamp and the Globus authentication mechanism [18] to protect each certificate. There are two potential protection strategies we will consider. Initially, we will publish price certificates in unprotected IBP buffers and we will implement a "redemption" interface through which an accounting system will be able to verify the authenticity of a certificate. First, the accounting system and a First Bank of G Branch Office will authenticate each other (via the Globus authentication protocol) and then the accounting system will be able to present a certificate so that the Branch Office can check its validity. Validity checking can either be via encrypted MD5 [30] checksum or by keeping protected shadow copies of each certificate under the Bank's control. IBP's time-limited buffer semantics can automatically time out, and reclaim the space for certificates that have expired.

A more sophisticated strategy is to generate a set of Globus authentication credentials each price certificate so that any holder of the certificate may check its validity using the authentication protocol. That is, any process may authenticate a price certificate with the First Bank of G (which will serve as the certificate authority) to check its validity via the Globus authentication mechanism itself. We will investigate the feasibility of using a user authentication mechanism as a way of managing non-forgable and protected data structures as part of this project.

4 Testing Computational Economies

The First Bank of G will provide a vehicle for deploying different computational economies in a variety of Grid settings. We plan to focus on experimental verification of our results in "live" Grid environments, although we will explore simulation as an additional verification mechanism.

We will use the SInRG (Scalable Intracampus Research Grid) research infrastructure at the University of Tennessee as the initial platform for g-Commerce research and verification. Recently funded by the National Science Foundation[2], the goal of SInRG is to deploy computational, network, and storage resources throughout the University of Tennessee, Knoxville campus for use as dynamically allocatable resources. Initial SInRG users include computational ecologists, computational biologists, medical imaging researchers, researchers studying material science, and researchers from electronic engineering. University of Tennessee Grid middleware (The NWS [38], IBP [27], NetSolve [7], EveryWare [36], and AppLeS [5, 32]) as well as other Grid infrastructures [15, 19, 33] will be deployed across all SInRG resources to support both Grid system and applications research. We will use this unique combination of user and research communities to study the effectiveness of g-Commerce as a resource allocation strategy. SInRG will provide our project both with a valuable source of resource supply and demand data and a live testbed for the First Bank of G. We will

- deploy the First Bank of G in the SInRG environment,
- instrument SInRG middleware and resources with demand and supply sensors (respectively),
- modify NetSolve and AppLeS scheduling agents to use price certificates in making scheduling decisions for application execution on SInRG, and
- deploy this new g-Commerce enabled middleware throughout the campus Grid for access by SInRG users.

Thus, we will leverage SInRG as the experimental test apparatus for the project. At the same time, once operational, the First Bank of G will provide an important service to the SInRG community.

The project will also take advantage of results generated by the Grid Application Development Software[3] (GrADS) project, funded through the National Science Foundation's Next Generation Software program. While the goal of GrADS is to provide a comprehensive Grid software environment, and g-Commerce will certainly play an important role in defining robust allocation and accounting strategies for GrADS, we plan to make use of the GrADS MicroGrid [8] system to provide a simulation environment for our work. The MicroGrid is a simulation tool-kit that is designed to enable repeatable simulation experiments using unmodified Grid applications and services. An application run on the MicroGrid has all of its Grid service requests intercepted by the MicroGrid infrastructure. Artificial load

[2] Both Investigators for this grant are Co-Investigators for SInRG
[3] Dr. Wolski is a Co-investigator on the GrADS project.

and performance response (based on either synthetic or performance trace data) can be induced and the application's behavior observed. We plan to make use of this powerful simulation environment to conduct repeatable experiments, and to test our economic policies under a wide range of conceivable performance conditions.

Finally, we will make the software artifacts that result from this project available to the broader community via the National Partnership for Advanced Computational Infrastructure (the PI is an NPACI partner). We will work with operations personnel from NPACI and the National Computational Science Alliance (NCSA) to harden the First Bank of G implementation, devise an appropriate economic policy, and integrate the First Bank of G with the extant accounting systems. Feedback from the partnerships will serve to hone and solidify the g-Commerce concepts. At the same time, g-Commerce will play an important role in helping to achieve NPACI and NCSA usage goals.

5 Conclusion

The vision of ubiquitous and seamless computational power available from distributed, heterogeneous resources has resulted in a new software architecture known as the *Computational Grid* [16]. Early research is fueling national-scale infrastructure and deployment of nascent but ever-maturing Computational Grid technology. Proof-of-concept applications and and suites of software demonstrate the promise of "the Grid," but there remain several serious research challenges that must be addressed before the vision becomes real and tangible. One such challenge is the problem of resource allocation.

Effective resource allocation strategies are problematic for many reasons. The Grid is intended to draw power from a resource collection that fluctuates dynamically. Resources are committed, either permanently or temporarily, to a Grid-wide pool by individual resource owners or managing institutions. These "stake holders" must maintain ultimate local control over the resources they manage, thus making a single, over-arching resource manager or resource management policy infeasible. The Grid cannot dictate how resources should be managed locally. On the other hand, there must be some structure placed on the allocation of resources in order to prevent Grid users from wantonly contending with each other for resources as they become available, which would result in system oscillation and instability.

The thesis of this proposal is that Grid resource allocation is best structured as a *market economy*, where prices are dynamically affixed to resources according to their supply and demand. We call term research *g-Commerce*, since we apply economic principles to Grid computing, as e-commerce applies economic principles to Internet computing. Resource prices are determined by a distributed pricing entity that we term *The First Bank of G*. This entity dynamically monitors resource supply and demand, and sets prices based on current levels so that supply matches demand (prices are fair) *and* the overall allocation of resources is stable. This combination of fair price determination and system-wide stability differentiates g-Commerce from auction-based approaches to the brokering of resources.

To restate: We propose to study, develop, implement and deploy economically based systems for allocating resources on the Computational Grid. The economic policies will be efficient to evaluate in distributed environments and will promote Grid-wide allocation stability. Our research plan to achieve these ambitious goals is as follows:

— Theory: We will study economic pricing models of market economies and apply them to the problem of Grid resource allocation. We begin with a powerful result by Smale, and will derive results concerning market stability, robustness in the presence of incomplete information, and tolerance to failure. We will then extend this research to study alternative models based on subjective valuation of resources, and on strategies for resource "packaging." Again, our goal will be to apply these models to Grid systems and to prove results about their stability and efficiency characteristics.

— Development: In the development phase, we start to build *The First Bank of G*. This Grid service is a distributed pricing entity that collects information from resource suppliers and users in order to determine the price of resources. We will build upon successful Grid tools for dynamic information collection and resource availability forecasting [35, 38, 25], and develop interfaces for suppliers and users to present their pricing parameters to the Bank. Once the information is collected and prices are determined, they will be made available to suppliers and users in the form of *price certificates*. These abstractions can be employed by automatic scheduling agents so that they may determine optimal or near optimal resource selections for their respective tasks.

— Testing and Verification: We will test our theory and pricing policies in two ways. First is by simulation, based on traces or on stochastic parameterization of resource supply and demand. We intend to employ Chien's recent

MicroGrid software [8] as a simulation engine. Second, we will test our principles on the University of Tennessee's Scalable Intracampus Research Grid (SInRG) [12]. SInRG is a federated collection of computing resources distributed on the campus of the University of Tennessee. The intent of SInRG is to provide a campus infrastructure testbed and computing platform that mirrors national Grid efforts. It, therefore, serves as a research infrastructure (complete with a supportive user community) within which middleware for Grid computing may be developed and tested using actual, real-world loads and resources. The g-Commerce effort will be a research project which both benefits from SInRG and, as our results emerge, ultimately contributes to it.

References

[1] The U.C. Berkeley Millenium Project, SIMS annual report, 1999. http://millennium.millennium.berkeley.edu/groups/GRP_

[2] AppLeS. http://www-cse.ucsd.edu/groups/hpcl/apples/apples.html.

[3] K. Arrow and L. Hurwicz. The stability of the competitive equilibrium i. *Econometrica*, 26:522–52, 1958.

[4] J. Basney and M. Livny. Improviong goodput by co-scheduling cpu and network capacity. *International Journal of High Performance Computing Applications*, 13, 1999.

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

[6] 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.

[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] A. Chien. Microgrid: Simulation tools for computational grid research. http://www-csag.ucsd.edu/projects/grid/microgrid.html.

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

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

[11] C. Czajkowski, I. Foster, and C. Kesselman. Co-allocation services for computational grids. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, 1999.

[12] J. J. Dongarra *et al.* Research infrastructure: The scalable intracampus research grid for computer science research. National Science Foundation, Grant EIA-9972889, http://www.nsf.gov/cgi-bin/showaward?award=9972889, Jan. 2000.

[13] W. Elwasif, J. S. Plank, M. Beck, and R. Wolski. *IBP-Mail*: Controlled delivery of large mail files. In *NetStore '99: Network Storage Symposium*. Internet2, October 1999.

[14] The everyware home page – http://nws.npaci.edu/everyware.

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

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

[17] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. A distributed resource management architecture that supports advanced reservation and co-allocation. In *International Workshop on Quality of Service*, 1999.

[18] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. Security architecture for computational grids. In *Proc. 5th ACM Conference on Computer and Communications Security Conference*, pages 83–92, 1998.

[19] 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.

[20] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, Sept. 1996.

[21] V. Jacobson. Congestion avoidance and control. In *Proceedings of SIGCOMM '88*, volume 18, August 1988.

[22] 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. available from http://www-fp.mcs.anl.gov/otc/Guide/TechReports/index.html.

[23] D. Mills, A. Thyagarajan, and B. Huffman. Internet timekeeping around the globe. In *Proc. Precision Time and Time Interval (PTTI) Applications and Planning Meeting*, 1997.

[24] 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.

[25] The network weather service home page – http://nws.npaci.edu.

[26] 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.

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

[28] J. S. Plank, M. Beck, W. Elwasif, T. Moore, M. Swany, and R. Wolski. The Internet Backplane Protocol: Storage in the network. In *NetStore 99: Network Storage Symposium*. Internet2, http://dsi.internet2.edu/netstore99, October 1999.

[29] R. L. Ribler, J. S. Vetter, H. Simitci, and D. A. Reed. Autopilot: Adaptive control of distributed applications. In *Proc. 7th IEEE Symp. on High Performance Distributed Computing*, Aug 1998.

[30] R. Rivest. The md5 message-digest algorithm, 1987. ARPA Working Group Requests for Comment DDN Network Information Center, SRI International, Menlo Park, CA, RFC-1321.

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

[32] N. Spring and R. Wolski. Application level scheduling: Gene sequence library comparison. In *Proceedings of ACM International Conference on Supercomputing 1998*, July 1998.

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

[34] 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.

[35] 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.

[36] R. Wolski, J. Brevik, C. Krintz, G. Obertelli, N. Spring, and A. Su. Running everyware on the computational grid. In to appear *SC99 Conference on High-performance Computing*, April 1999. available from `http://www.cs.utk.edu/~rich/papers/ev-sc99.ps.gz`.

[37] R. Wolski, J. Brevik, C. Krintz, G. Obertelli, N. Spring, and A. Su. Writing programs that run everyware on the computational grid. Technical Report UT-CS-99-420, The University of Tennessee, April 1999. available from `http://www.cs.utk.edu/~rich/papers/ev-results.ps.gz`.

[38] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems (to appear)*, 1999. available from `http://www.cs.utk.edu/~rich/publications/nws-arch.ps.gz`.

[39] R. Wolski, N. Spring, and J. Hayes. Predicting the cpu availability of time-shared unix systems on the computational grid. In *Proc. 8th IEEE Symp. on High Performance Distributed Computing*, 1999. available from `http://www.cs.utk.edu/~rich/publications/nws-cpu.ps.gz`.