A Case for Economy Grid Architecture for Service Oriented Grid Computing

Rajkumar Buyya, David Abramson, and Jonathan Giddy

School of Computer Science and Software Engineering
Monash University
Caulfield Campus, Melbourne, Australia

CRC for Enterprise Distributed Systems Technology
Monash University
Caulfield Campus, Melbourne, Australia

Email: {rajkumar, davida, jon}@csse.monash.edu.au

Abstract

Computational Grids are a promising platform for executing large-scale resource intensive applications. However, resource management and scheduling in the Grid environment is a complex undertaking as resources are (geographically) distributed, heterogeneous in nature, owned by different individuals or organizations with their own policies, have different access and cost models, and have dynamically varying loads and availability. This introduces a number of challenging issues such as site autonomy, heterogeneous interaction, policy extensibility, resource allocation or co-allocation, online control, scalability, transparency, resource brokering, and "computational economy".

A number of Grid systems (such as Globus and Legion) have addressed many of these issues with exception of a computational economy. We argue that a computational economy is required in order to create a real world scalable Grid because it provides a mechanism for regulating the Grid resources demand and supply. It offers incentive for resource owners to be part of the Grid and encourages consumers to optimally utilize resources and balance timeframe and access costs. We propose a 'computational economy framework' that builds on the existing Grid middleware systems and offers an infrastructure for resource management and trading in the Grid environment. We discuss the usage economic models for resource trading in the Nimrod/G resource broker and present deadline and cost-based scheduling experimental results on the Grid.

1. Introduction

Grid [13] based computational infrastructure is a promising next generation computing platform for solving large-scale resource intensive problems. It couples a wide variety of geographically distributed computational resources (such as PCs, workstations, and clusters), storage systems, data sources, databases, computational kernels, and special purpose scientific instruments and presents them as a unified integrated resource. However, including application development, the management and scheduling of computations in the Grid environment is a complex undertaking as resources are geographically distributed, heterogeneous in nature, owned by different individuals or organizations with their own policies, have different access cost models with dynamically varying loads and availability conditions. A typical market-oriented Grid environment is shown in Figure 1. It encompasses a wide range of software technologies from local operating environments (operating or queuing systems) to global resource brokers and applications that are designed to exploit Grid capability. The interactions between these components must be secure and adapt to the changing resource status. Internationally, there are many projects [2][16] actively exploring the design and development of different Grid system components and services for secure execution of applications on wide-area resources.

As shown in Figure 1, the users in global Grid environment essentially interact with a Grid Resource Broker (GRB) that hides the complexity of resource management and scheduling. The broker discovers resources using Grid Information Services (GIS), negotiates with grid-enabled resources or their agents for service costs, performs resource selection, maps and schedules tasks to resources, stages the application and data for processing on remote resources, and finally gathers results and hands them to the user. It is also responsible for monitoring application execution progress along with managing and adapting to changes in the Grid environment such as resource failures.

In this paper we identify requirements of users (resource providers and consumers) in the Grid economy and various resource management issues that need to be addressed in realizing such a Grid system. We briefly discuss popular economic models for resource trading and present related work that employs computational economy in resource management. We propose a scalable architecture and new services for the

Grid that provide mechanisms for addressing user requirements. The proposed architecture leverages services offered by the existing Grid systems such as Globus and offers new core services for resource trading. We discuss the use of these economic models and services for developing tools such as the Nimrod/G resource broker. We discuss a case study consisting of scheduling a parameter-sweep application on a large computational Grid spanning four continents and present some experimental results.

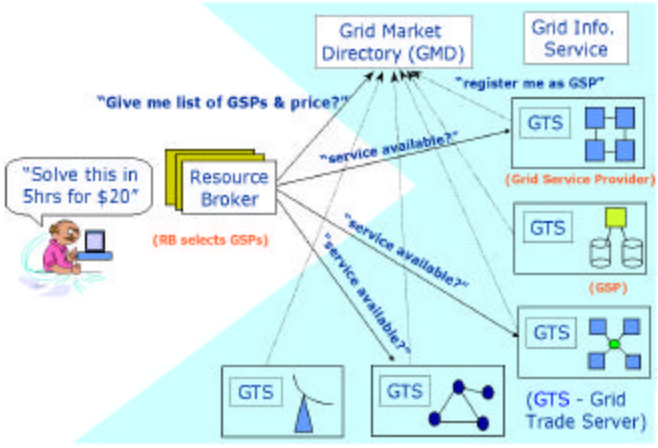


Figure 1: A Generic view of interaction between players in an Economy Grid.

2. Grid Economy and Resource Management Issues

The current research and investment into computational grids is motivated by an assumption that coordinated access to diverse and geographically distributed resources is valuable. In this paradigm, we need mechanisms that allow such coordinated access, but also sustainable, scalable models and policies that promote precious Grid resource sharing. Based on the success of economic institutions in the real world as a sustainable model for exchanging and regulating resources, goods and services, we propose a computational economy framework. Among other things, this framework provides a mechanism to indicate which users should receive priority. In [4], we have presented several arguments in favor of developing Grid architecture for computational economy and its benefits.

Like all systems involving goals, resources, and actions, computations can be viewed in economic terms [51]. With the proliferation of networks, high-end computing systems architecture has moved from centralized toward decentralized models of control and action; use of economic driven market mechanisms would be a natural extension of this development. The ability of trade and price mechanisms to combine local decisions by diverse entities into globally effective characteristic implies their value for organizing computations in large systems such as Internet scale computations Grids.

The two key players in market oriented computational Grid are *resource providers* (we refer hereafter as GSPs—Grid Service Providers) and *resource consumers* (we refer hereafter as GRBs—Grid Resource Broker that acts as a consumer's software agent). Both have their own expectations and strategies for being part of the Grid. In this Grid economy, resource consumers adopt the strategy of solving their problems at low cost within a required timeframe and resource providers adopt the strategy of obtaining best possible return on their investment. The resource owners try to maximize their resource utilization by offering a competitive service access cost in order to attract consumers. The users (resource consumers) have an option of choosing the providers that best meet their requirements. If resource providers have local users, they will try to recoup the best possible return on "idle/leftover" resources. In order to achieve this, the Grid systems need to offer tools and mechanisms that allow both resource providers and consumers to express their requirements. The Grid resource consumers interact with brokers (also called super-schedulers) to express their requirements such as the *budget* that they are willing invest for solving a given problem and a *deadline*, a timeframe by which they need results. They also need capability to trade between these two requirements and steer the computations accordingly. The Grid Service Providers need tools for expressing their pricing policies and mechanisms that help them to maximize the profit and resource utilization. Various economic models, ranging from commodity market to auction-based, can be adopted for deciding pricing strategies. The Grid infrastructure needs to support these economic models for resource trading.

To date, individuals or organizations that have contributed resources to the Grid have been largely motivated by the public good, prizes, fun, fame, or collaborative advantage. This is clearly evident from the construction of private grids (but on volunteer resources) or research test-beds such as Distributed.net [9], SETI@Home [20], Condor pool [7], DAS (Distributed ASCI Supercomputer) [10], GUSTO [14], and eGrid [11]. Even commercial companies such as Entropia, ProcessTree, Popular Power, Mojo Nation, United Devices, and Parabon are exploiting idle CPU cycles from desktop machines to build a commercial computational Grid [16]. These companies are able to develop large-scale infrastructure for Internet computing and use it for their own financial gain by charging for access to CPU cycles for their customers without

offering fiscal incentive to all resource contributors. In the long run, this model is less likely to succeed in creating a maintainable and sustainable infrastructure. Therefore, a Grid economy seems a better for model for managing and handling requirements of both Grid providers and consumers. It is interesting to note that, even in electricity Grid, bid-based electricity trading over the Internet has been adopted to develop competitive forces in the electricity marketplace [27].

An economy approach to grid computing introduces a number of new issues to be addressed in addition to those already addressed by existing Grid systems. Grid toolkits such as Globus [8] have addressed the five challenging resource management problems introduced by computational grids: site autonomy, heterogeneous substrate, policy extensibility, resource allocation or co-allocation, and online control. In [4], we proposed a "computational economy" as another key challenging issue that needs to be addressed for developing a service oriented Grid. We proposed an economy-based resource management architecture called GRACE (GRid Architecture for Computational Economy). The GRACE architecture is designed in such a way that it reuses or leverages services supported by the existing infrastructures (such as Globus [12], Legion [18], Condor/G [7], QBank [37], and NetCash [39]) as much as possible. It offers new services that are particularly missing in them for constructing an economy Grid. The economy Grid framework needs to provide infrastructure that offers the following:

- An Information and Market directory for publicizing Grid entities
- Models for establishing the value of resources
- Resource pricing schemes and publishing mechanisms
- Economic models and negotiation protocols
- Mediators to act as a regulatory agency for establishing resource value, currency standards, and crisis handling.
- Accounting, Billing, and Payment Mechanisms

3. Economy Models and Related Work

A market-based approach in computational system design has been the topic of research over the years [19][26][28][40]. Some of these systems have developed a substantial theoretical foundation but without large-scale deployment, experimental validation, and testing. A number of recent systems are attempting to apply computational economy for Web-based computing or for cluster-based systems. Research in the area of artificial intelligence and agents based computing has explored economy-based approach for migration of agents and resource allocation. FIPA (Foundation for Intelligent Physical Agents), a consortium of the software agents community, has proposed a specification for agents negotiation [24].

Various economic models for resource trading and establishing pricing strategies have been proposed [6][19][27][29][41][42] and they include,

- A Commodity Market (Flat or Demand & Supply driven pricing) Model
- A Posted Price Model
- A Bargaining Model
- A Tendering/Contract-Net Model
- An Auction Model
- A Bid-based Proportional Resource Sharing Model
- A Community/Coalition/Bartering Model

In [6], we presented architecture and issues associated in implementing the above economic models in the Grid environment. In the context of business negotiation on the Internet these models have been discussed in [23]. The resource providers and consumers can use any one or more of these economic models or even a combination of them while establishing access price depending upon their objective functions. Either the GSP or the GRB can initiate resource trading and participate in a market like environment depending on their requirements. In the *commodity market* model, resource providers competitively set the price and advertise their service in business directory as service providers (see Figure 1). The pricing scheme can be static or dynamic in nature. Consumers choose resource providers through cost-benefit analysis. The *posted price* model is similar to commodity market model except that it posts offers long before scheduling. In the *bargaining model*, providers and consumers negotiate for resource access cost and time that maximizes their objectives. In these three models other consumers do not influence the price for access to services. The negotiation happens privately between a consumer and a provider and there is no way for a consumer to know how much others value the resource services. Accordingly, the consumers need to decide whether to accept/reject offer depending on its private objective function.

In the *Tender/Contract Net* model, the consumer (GRB) invites sealed bids from several GSPs and selects those bids that offer lowest service cost within their deadline and budget. In the *Auction* model, producers invite bids from many consumers and each bidder is free to raise their bid accordingly. The auction ends when no new bids are received. The auction can be performed through open or closed bidding protocols. In the *bid-based proportional resource-sharing* model, the amount of resource allocated to consumers is proportional to the

value of their bids.

In the *Community/Coalition/Bartering* model, a group of individuals can create a cooperative computing environment to share each other's resources. Those who are contributing resources to a common pool can get access to resources when in need. A sophisticated model can also be employed for deciding the share of resources a contributor can obtain and can allow a user to accumulate credit for future needs. Systems like Mojonation.net employ this credit-based bartering model for storage sharing across the community network. This model works when all participants in the Grid are both service providers and consumers.

Several research systems (see Table 1) have attempted to apply the concept of computational economy for information management, CPU cycles, Storage, and Network access. They include Mariposa [19], Mungi [30], Popcorn [33], Java Market [31], Enhanced MOSIX [32], JaWS [17], Xenoservers [34], D'Agents [35], Rexec/Anemone [29], Spawn [36], Mojo Nation [25]. These systems can manage either single or multiple resources and they are categorized as follows:

- Single Domain Systems: Enhanced MOSIX and Rexec/Anemone.
- Web-based Systems: Popcorn, Java Market, and JaWS.
- Agent-based systems: Xenoservers and D'Agents.
- Database/Storage systems: Mariposa and Mungi.

Each of the resource management systems discussed in Table 1 follows a single model for resource trading. They have been designed with a specific goal in mind either for CPU or storage management. In order to use some of these systems, applications have to be designed using their proprietary programming models, which is generally discouraging, as applications need to be specifically developed for executing on those systems. Also, resource trading and job management modules have been developed as integrated monolithic systems. In our system, we have separated these two concerns through layered design approach. The resource trading services are offered as core services and they can be used by higher-level services/tools such as resource brokers. Another key advantage of our system (a combination of GRACE and Nimrod/G) is that it allows the execution of legacy applications on large wide-area distributed systems.

4. Grid Architecture for Computational Economy (GRACE)

A computational economy-based architecture for Grid resource management is shown in Figure 2. We generally refer to this architecture as GRACE (Grid Architecture for Computational Economy). This architecture is generic enough to accommodate different economic models used for resource trading for determining the service access cost. The key components of Grid include,

- User Applications (sequential, parametric, parallel, or collaborative applications)
- Higher Level Services and Tools (such as Resource Brokers or Market oriented programming environments)
- Middleware (services resource trading and coupling distributed wide area resources)
- Grid Fabric components including local resource managers (e.g., Queuing systems)

As mentioned earlier, our goal is to realize this economy Grid architecture by leveraging existing infrastructures such as Globus/Legion as much as possible and develop new services that are particularly missing in them. Therefore, we mainly focus on two things: first, develop middleware services for resource trading using different economic models, second use these services along with other middleware services in developing advanced user-centric Grid resource brokers. Throughout this section, we discuss how we are realizing our economy Grid vision and show co-existence of our modules with other systems.

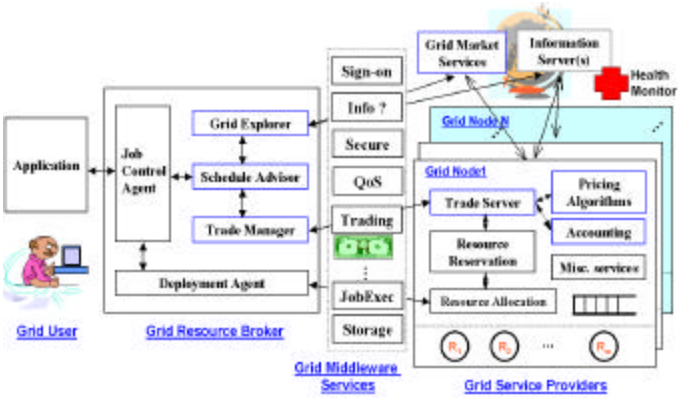


Figure 2: Abstract Grid Architecture

GRACE provides services that help resource owners and user-agents maximize their objective functions. The resource providers can contribute their resource to the Grid and charge for services. They can use GRACE mechanisms to define their charging and access policies and the GRACE resource trader works according to those policies. The users interact with the Grid by defining their requirements through high-level tools such as resource brokers (also known as superschedulers or metaschedulers). The resource brokers work for the consumers and attempts to maximize user utility. They can use GRACE services for resource trading and identifying GSPs that meets its requirements.

4.1 Grid Resource Broker (GRB)

The resource broker acts as a mediator between the user and grid resources using middleware services. It is responsible for resource discovery, resource selection, binding of software, data, and hardware resources, initiating computations, adapting to the changes in grid resources and presenting the grid to the user as a single, unified resource. The components of resource broker are the following:

- **Job Control Agent (JCA):** This is a persistent control engine responsible for shepherding a job through the system. It coordinates with schedule adviser for schedule generation, handles actual creation of jobs, maintenance of job status, interacting with clients/users, schedule advisor, and dispatcher.
- **Schedule Advisor (Scheduler):** This is responsible for resource discovery (using grid explorer), resource selection and job assignment (schedule generation) so as to ensure that the user requirements are meet.
- **Grid Explorer:** This is responsible for resource discovery by interacting with grid-information server and identifying the list of authorized machines, and keeping track of resource status information.
- **Trade Manager (TM):** This works under the direction of resource selection algorithm (schedule advisor) to identify resource access costs. It uses market directory services and GRACE negotiation services for trading with grid service provides (i.e., their representative trade servers).
- **Deployment Agent (DA):** It is responsible for activating task execution on the selected resource as per the scheduler's instruction and periodically update the status of task execution to JCA.

Our Nimrod/G broker has components that support similar functions.

4.2 Economy Grid Middleware in Globus Context

The grid middleware offers services that help in coupling a grid user and remote resources through a resource broker or grid enabled application. It offers core services [2] such as remote process management, co-allocation of resources, storage access, directory information, security, authentication, and Quality of Service (QoS) such as resource reservation for guaranteed availability and trading for minimizing computational cost. Many of these services are already offered by Globus [12] components and they include,

- Resource allocation and process management (GRAM).
- Resource Co-allocation services (DUROC)
- Unicast and multicast communications services (Nexus)
- Authentication and related security services (GSI)
- Distributed access to structure and state information (MDS)
- Status and Health Monitoring components (HBM)
- Remote access to data via sequential and parallel interfaces (GASS)
- Construction, caching, and location of executables (GEM)
- Advanced resource reservation (GARA)

We provide components (see Figure 3) that offer services required for constructing economy Grid and that can co-exist with systems like Globus:

- A resource broker (e.g., Nimrod/G)
- Various resource trading protocols
- A mediator for negotiating between users and grid service providers (Grid Market Directory)
- A deal template for specifying resource requirements and services offers
- A trade server
- A pricing policy specification
- Accounting (e.g., QBank) and payment management (GBank)

The new middleware services being proposed are designed to offer low-level services that co-exist with Globus services and infrastructure. These core services can be used by higher level services and tools such as the Nimrod/G Resource Broker that can use various economic models suitable for meeting user requirements.

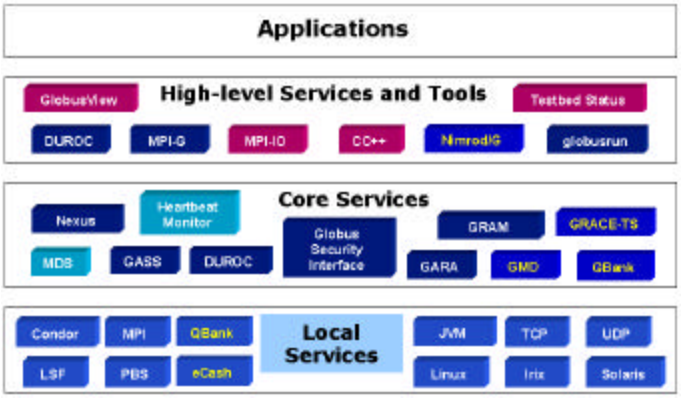


Figure 3: Economy Grid Components within Globus context.

The Grid service providers specifically deal with the following components along with Globus components:

- **Trade Server (TS)**: This is a resource owner agent that negotiates with resource users and sells access

to resources. It aims to maximize the resource utility and profit for its owner i.e., earn as much money as possible. It consults pricing policies during negotiation and directs the accounting system for recording resource consumption and billing the user according to the agreed pricing policy.

- **Pricing Policies**: These define the prices that resource owners would like to charge users. The resource owners may follow various policies to maximise their profit and resource utilisation and the price they charge may vary from time to time and one user to another user. The pricing can also be driven by demand and supply like in the real market environment. That is, in this commodity market model, pricing is essentially determined by objective functions of service providers and users. The pricing policy can also be based on auction. In this auction based economic model, pricing is driven by how much users value for the service and the highest bidder wins the access to Grid services.
- **Resource Accounting and Charging** components (such as GBank along with QBank) are responsible for recording resource usage and bills the user as per the usage agreement between resource broker (TM, a user agent) and trade server (resource owner agent).

4.3 Grid Open Trading Protocols and Deal Template

The resource trading protocols define the rules and format for exchanging commands between a GRACE client (Trade Manager) and Trade Server. Figure 4 shows a finite state machine representation of multilevel negotiation protocols that both client and server need to follow for the bargaining/tender model. In this model, the Trade Manager (TMs) contacts Trade Server (TSs) with a request for a quote. The TM specifies resource requirements in a Deal Template (DT), which can be represented by a simple structure with its fields corresponding to deal items or by a "Deal Template Specification Language", similar to the *ClassAds* mechanism employed by the Condor [7] system. The contents of DT include, CPU time units, expected usage duration, storage requirements along with its initial offer. The TM looks into DT and updates its contents and sends back to TS. This negotiation between TM and TS continues until one of them indicates that its offer is final. Following this, the other party decides whether to accept or reject the deal. If accepted, then both work as per the agreement mentioned in the deal. The overhead introduced by the multilevel point-to-point protocol can be reduced when resource access prices are announced through grid information services (e.g., MDS) or market directory.

Some interaction protocols for a business negotiation on the Internet have been presented in [23]. This paper highlights some commonalties in the structure of different price negotiation mechanisms such as fixed price sales, auctions, and brokerages. These business negotiation models and protocols are also applicable for our resource trading and we have already explored such models and protocols in our resource management and scheduling system.

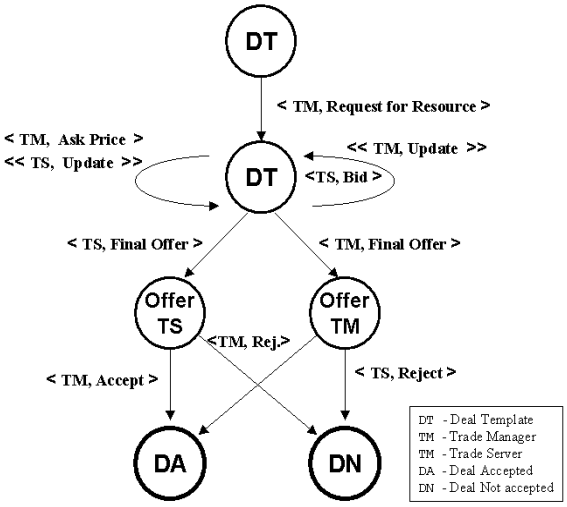


Figure 4: A finite state representation of resource trading negotiation (for market/bargain model).

4.4 Pricing, Accounting, and Payment Mechanisms

In a computational economy Grid environment, both resource owners and users want to maximize their benefits. As there will be many GSPs offering similar services, they need to have competitive pricing structure in order to attract users, efficiently utilize resources, and maximize profit. The resources consumed by the user applications need to be accounted and charged. Various payment mechanisms need to be supported. The users can purchase resource access credits in advance or pay-after-usage. Each GSP can maintain this by using systems like QBank or there can be global Grid-wide bank called that mediates payment for services accessed by the user. Figure 5 shows various components at GSP node and their interactions during resource trading, consumption, metering (measuring), billing, and payment handling.

How to determine the Price?

A simple pricing scheme is a fixed price model, but this does not work when the users demand QoS. This

requirement changes between applications and across time. The demand/supply and QoS driven pricing schemes have been investigated by many researchers in the context of software Agents [21][22]. The pricing schemes based on various parameters can be developed and they include,

- A flat price model (the same cost for applications and no QoS like in today's Internet [44])
- Usage timing (peak, off-peak, lunch time like pricing telephone services)
- Usage period and duration (short/long)
- Demand and supply (e.g., Smale model [46])
- Foresight-based [21] (i.e., an ability to model and predict responses by competitors)
- Loyalty of Customers (like Airlines favoring frequent flyers!)
- Historical data
- Advance agreement/contract with service provides
- Calendar based
- Bulk Purchase
- Voting in which trade unions decide pricing structure
- Resource capability as benchmarked in the capital market
- Application areas in which academic R&D or public good applications can be offered at cheaper rate compared to commercial applications.

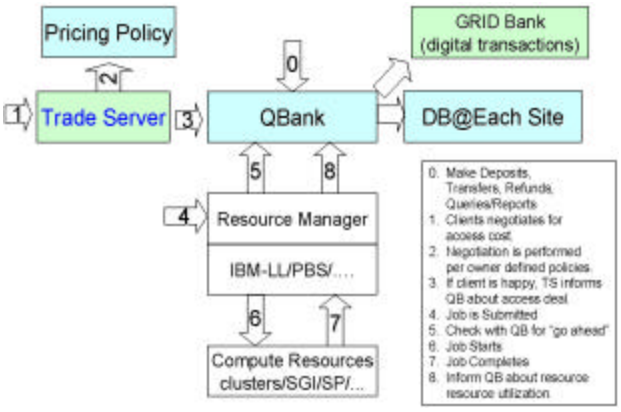


Figure 5: An Interaction between GSP resource management components.

In [22], five different provider pricing strategies, ranging widely from ones that require perfect knowledge and unlimited computational power to ones that require very little knowledge or computational capability, are employed in two different buyer populations, namely quality-sensitive and price sensitive buyers. The resulting collective dynamics have been investigated using a combination of analysis and simulation. In a population of *quality-sensitive buyers*, all pricing strategies lead to a price equilibrium predicted by a game-theoretic analysis. However, in a population of *price-sensitive buyers*, most pricing strategies lead to large-amplitude cyclical price wars. These pricing strategies and issues are also applicable to the Grid and strategies need to be designed such that the resource providers benefit through efficient resource utilization and consumers will have the ability to trade-off between cost and timeframe in the Grid marketplace.

Service items to be Charged and Accounted

User applications have different resource requirements depending on computations performed and algorithms used in solving problems. Some applications can be CPU intensive while others can be I/O intensive or a combination. For example, in CPU intensive applications it may be sufficient to charge only for CPU time whilst offering free I/O operations. This scheme cannot be applied for I/O intensive applications. Therefore, consumption of the following resources need to be accounted and charged:

- CPU - User time (consumed by user App.) and System time (consumed while serving user App.)
- Memory
- Maximum resident set size - page size
- Amount of memory used
- Page faults
- Storage used
- Network activity
- Signals received, context switches
- Software and Libraries accessed (particularly required for the emerging ASP world).

Access to each these entities can be charged individually or in combination. Combined pricing schemes need to have a costing matrix that takes a request for multiple resources in pricing. An economic model proposed by Smale [46] allows formulation of such pricing schemes for resource allocation.

Payment Mechanisms

A computational economy Grid needs to support various payment mechanisms. They include:

- Prepaid – Pay and use in which users need to buy credits in advance from GSPs or Grid Bank
- Use and pay later
- Pay as you go
- Grants based

Each GSP can bill their users directly and handle all payment processing issues themselves. This method introduces a great burden for both providers and users in a large-scale Grid environment. This can be simplified by having mediators like a Grid-wide Bank. The users can inform GSPs about their Grid Bank account details for which they can charge directly or users can pay by other electronic cash systems. This can be achieved by

using digital currency mechanisms such as:

- NetCheque: [38] - Users registered with NetCheque accounting servers can write electronic cheques and send them to service providers. When deposited, the balance is transferred from sender to receiver account automatically.
- NetCash [39] - This supports anonymity and it uses the NetCheque system to clear payments between currency servers.
- Paypal [47] – This is an example of credit-card based automated mediator for payments processing.

Such electronic payment mechanisms satisfy the diverse requirements of service providers and their users. We are still investigating mechanisms for integrating such systems in our economy grid infrastructure.

4.5 System Prototype & Demo Experience

A prototype implementation of the Nimrod/G resource brokering and trading mechanisms has been demonstrated during the HPDC 2000 conference in Pittsburgh. In this parameter study experiment, we have been able to perform scheduling of parametric computations over Grid resources in Australia (Monash University Linux cluster and Solaris workstation) and the United States (DOE Argonne National Laboratories SGI/IRIX, IBM SP2, and Sun HPC machines, USC/ISI SGI-IRIX machine, and University of Virginia Linux cluster). These Unix-class HPC machines were Grid enabled by using Globus, Legion, and Condor/G system services. The modular or component-oriented architecture of Nimrod/G [3] resource broker allowed us to develop mechanisms for scheduling computations over resources enabled by these Grid middleware services (with minimal effort). The users prepare their application for parameter studies using Nimrod as usual [1]. The resulting parameter-sweep application can be executed on the Grid by submitting it to the Nimrod/G engine that mediates between the scheduler and deployment modules. The Nimrod/G scheduler uses directory services like the Globus MDS for resource discovery and the GRACE trading services for establishing an access price. Depending on the user preferences such as deadline, budget, and optimization parameters, Nimrod selects the best scheduling algorithm [5] for generating the schedule and assigning jobs to suitable resources. The Deployment Agent (DA) selects the right service module (Globus GASS/GEM/GRAM, Legion, or Condor/G) depending on the resource type for staging job/application and data on (remote) Grid resources, initiate computations and monitor their progress. As the performance of the Grid resources is not static, Nimrod/G performs rescheduling when scheduling event is raised. When job execution is finished, the DA gathers results from resources to the user space. During HPDC 2000 Demo, we started an experiment on our Solaris workstation in Australia from Pittsburgh and connected to the Nimrod/G engine for computational monitoring and steering. Using this remote steering client, we have been able to change deadline and budget to trade-off cost vs. timeframe for online demonstration of Grid marketplace dynamics.

Nimrod/G keeps record of all resource utilization and agreed pricing for resource access for accounting purpose. This information is useful for resource consumers for computational steering and verifying discrepancies in GSP billing statement and the actual amount of consumption. Resource provider can keep a record of resource consumption and bill/charge the user according to the agreed pricing.

5. Resource Trading and Scheduling Experimentation

In our previous experiments (performed on GUSTO test bed in 1999 [1]), the resource prices were hardwired into a file owned by the user. It was up to the user to ensure the prices in this file reflected the actual prices for each resource, and they were fixed for the entire duration of the experiment. This meant that the user needed to set the price to the highest price for a resource to ensure that their budget was not exceeded. This limitation is overcome by using GRACE resource-trading services. The Nimrod/G scheduler has been modified to use services of Grid Trade Servers running on each (gatekeeper) resource for establishing resource/service price. In order to test the operation of the Grid Trade Server, we performed an experiment by implementing the Posted Price Market Model for the Nimrod/G resource broker. A Grid testbed (shown in Figure 6) containing computational resources across four continents has been used in this experiment.

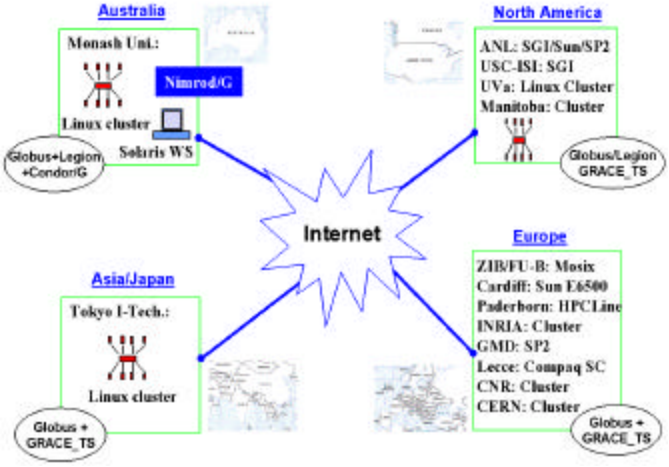
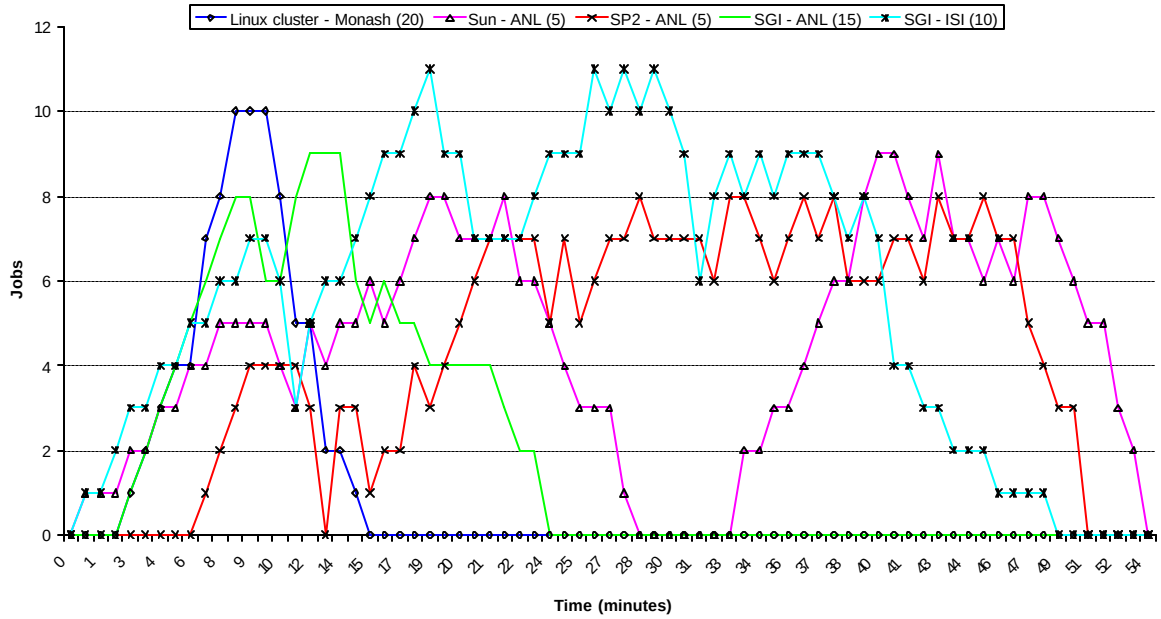


Figure 6: Global Economy Grid (EcoGrid) Testbed.

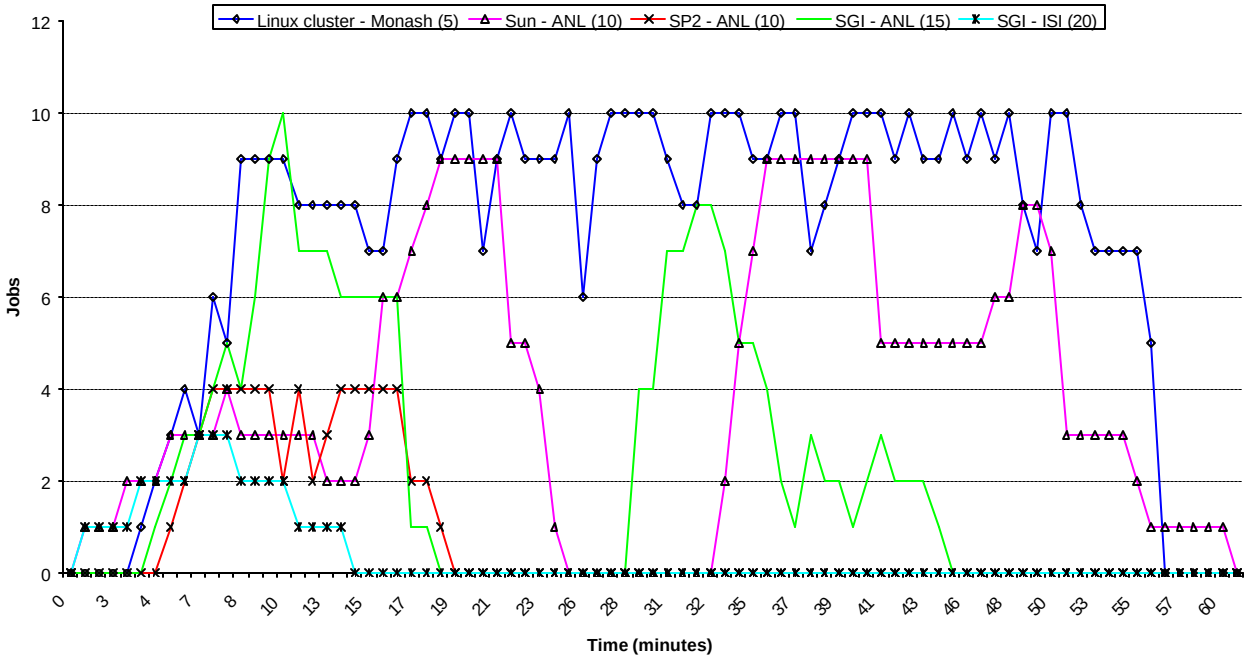
Resources selected for this experiment from the testbed are shown in Table 2. To test the Grid Trade Server with the current scheduler, we ran an experiment entirely during peak time and the same experiment entirely during off-peak time. It is important to note access price variations during peak and off-peak times and also time difference between Australia and US. The access price is expressed in Grid units (G$) per CPU second.

We selected 5 systems (shown in Table 2) from the testbed, each effectively having 10 nodes available for our experiment. Monash University has a 60-processor Linux cluster running Condor, which was reduced to 10 available processors for the experiment. Similarly, a 96-node SGI at Argonne National Laboratory (ANL) was made to provide 10 nodes by using *Condor glidein* to add 10 processors to the Condor pool. An 8-node Sun at Argonne and a 10-node SGI at the Information Sciences

Graph 1: Computational Scheduling during Australian Peak (or US off-peak) Time.



Graph 2: Computational Scheduling during Australian Off-peak (or US peak) Time.



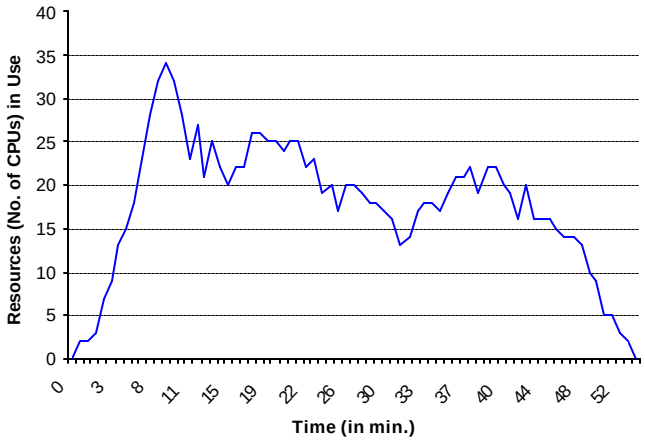
Institute (ISI) of the University of Southern California were accessed using Globus directly. Argonne's 80-node SP2 was also accessed directly through Globus. We relied on its high workload to limit the number of nodes available to us. We assigned artificial-cost (access price per second) for each of those resources depending on their relative capability. This is achieved by setting resource cost database, which is maintained on each of the resources by their owners. The resource cost database contains access cost (price) that they like to charge to all their grid users at different times of the day. The access price is generally various from users to users and time to time. The resource broker negotiates with trading servers for establishing access price using resource trading services (APIs) provided by the GRACE infrastructure.

We performed an experiment of 165 jobs. Each job was a CPU-intensive task of approximately 5 minutes duration. The experiment was run twice, once during the Australian peak time, when the US machines were in their off-peak times, and again during the US peak, when the Australian machine was off-peak. The experiments were configured to *minimise the cost*, within *one-hour deadline*. This requirement instructs the Nimrod/G broker to use "Cost-Optimization Scheduling algorithm" [5] in scheduling over the grid.

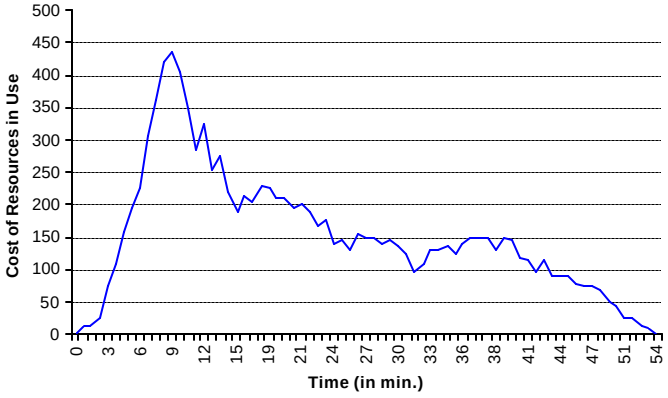
The number of jobs in execution/queued on resources (Y-axis) at different times (X-axis) during the experimentation is shown in Graph 1 and Graph 2. The results for the Australian peak experiment (Graph 1) show the expected typical results. After an initial calibration phase, the jobs were distributed to the cheapest machines for the remainder of the experiment. This characteristic of the scheduler is clearly visible in the Graphs 1 and 2. During Australian peak experiment, after calibration period, the scheduler excluded the usage of Australian resources as they were expenses and scheduler predicted that it could still meet the deadline using cheaper resources from US resources, which are in off-peak time phase. However, during Australia off-peak experiment, the scheduler never excluded the usage of Australian resources and in fact, it excluded the usage of some of the US resources as they were expensive at that time due to US in peak-time phase and their resources were expensive comparatively. The results for the US peak experiment (Graph 2) are somewhat more interesting. When the Sun becomes temporarily unavailable, the SP2, at the same cost, was also busy, so a more expensive SGI is used to keep the experiment on track to complete before the deadline.

When the scheduling algorithm tries to minimize the cost, the total cost Australian peak time experiment is 471205 units and the off-peak time is 427155 units. The result is that costs were quite low in both cases. An experiment using all resources without the cost optimization algorithm during the Australian peak cost 686960 units for the same workload. The cost difference indicates a saving in computational cost and it is certainly a successful measure of our budget and deadline-driven scheduling on the grid.

Graph 3: No. of Resources in use @ AU Peaktime.



Graph 4: Cost of Resources in use @AU Peaktime.

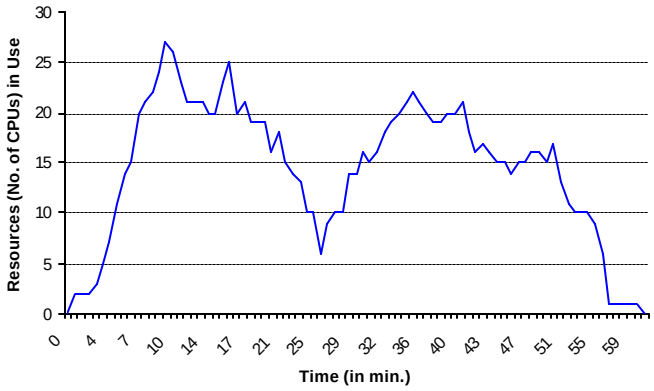


The number of computational nodes (CPUs) in use at different times during the execution of scheduling experimentation at Australian peak-time is shown in Graph 3. It can be observed that in the beginning of the experiment (calibration phase), scheduler had no precise information related to job consumption rate for resources, hence it tried to use as many resources as possible to ensure that it can meet deadline. After calibration phase, scheduler predicated that it could meet the deadline with fewer resources and stopped using more expensive nodes. However, whenever scheduler senses difficulty in meeting the deadline by using the resources currently in use, it includes additional resources. This process continues until

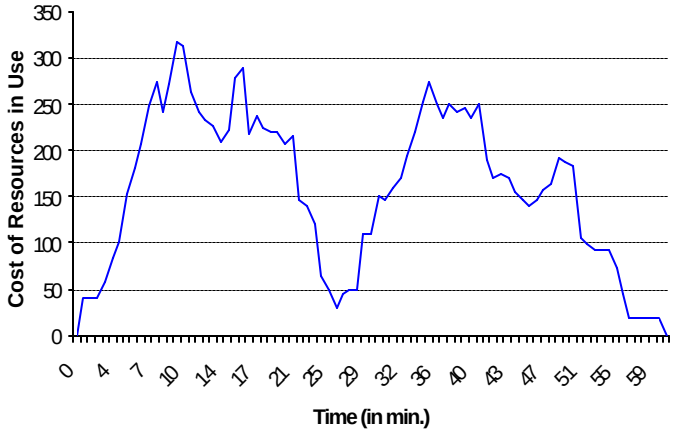
deadline is meet and at the same time it ensures that the cost of computation is within a given budget.

The total cost of resource (sum of the access price for all resources) in use at different times during the execution of scheduling experimentation at Australian peak-time is shown in Graph 3. It is interesting to observe the pattern of variation of cost during calibration phase is similar to that of number of resources in use. However, this is not the same as experiment progresses and in fact the cost of resources decreases almost linearly even thought resources in use does not decline at that rate. The reason for this behavior is that a large number of resources that the scheduler selected were from off-peaktime zone (i.e., US was in off-peak time when Australia was in Peak hours) as they were cheaper. Another reason is that the EconomyGrid testbed contains more US resources compared to Australian resources.

Graph 5: No. of Resources in use @ AU Off-peaktime.



Graph 6: Cost of Resources in use @AU Off-Peaktime.



Similar behavior did not occur in scheduling experiment conducted during Australian off-peak time (see Graph 5 and 6). The variation pattern of total number of resources in use and their total cost is similar due to the fact that the larger numbers of US resources were available cheaply. Although the scheduler has used Australian resources throughout the experiment (see Graph 2), the scheduler had to depend on US resources to ensure that the deadline is meet even if resources were expensive.

6. Conclusion and Future Work

We have discussed motivations for creating an economy Grid and issues involved in the development of resource management systems driven by computational economy. We proposed Grid Architecture for Computational Economy (GRACE) framework that takes benefits of existing middleware services and tools and offers new services that are essential for realizing a real world Grid. We briefly discussed economy models for resource trading in the Grid. They include commodity market, posted prices, bargaining, tendering, auction, proportional resource sharing or shareholder, and community/coalition/bartering models. We discussed the usage of resource trading in Grid brokering for posted price economic model and presented preliminary experimental results. The computational economics driven brokering system can be applied to peer-to-peer computing [48] applications that enable content sharing. Systems like Napster[49] or Gnutella [50] could use infrastructure that is similar to GRACE for encouraging people to share files, contents, or music in larger scale by providing them economic incentive. The brokering systems like Nimrod/G can discover the best content provider that meets consumers QoS requirements.

A computational economy approach for Grid resource management requires extensive exploration. For example, currently our Nimrod/G scheduler does not allow changes in the price of resources once initial scheduling decisions are made. That is, in scheduling the remaining jobs over the resources within the remaining budget, the scheduler makes significant assumptions about the future price of the resources. In addition, the scheduler uses the current price to calculate the cost of jobs that have completed in the past. Hence, using the current scheduler in a system where price varies over time makes the cost estimations meaningless, and the budget cannot be guaranteed. In order to overcome this limitation, we are currently investigating new scheduling algorithms that not only adapt to dynamic changing in resource conditions during runtime, but also to changes to access prices even during the execution of jobs. We will also be investigating new economic models such Auctions and Contract Net protocols for resource allocation. We

expect that economy driven approach to resource management and scheduling will make a great impact on the eventual success and widespread adoption of the Grid in day-to-day computational activities.

Acknowledgements

The work is funded through Monash University, Distributed Systems Technology Centre (DSTC), IEEE Computer Society, Centre for Distributed Systems and Software Engineering (DSSE), and the Australian Government research grants and scholarships. The computational resources that are part of the EconomyGrid testbed are owned or provided by Monash University, Argonne National Laboratories (USA), University of Southern California's Information Sciences Institute (USA), University of Virginia (USA), Tokyo Institute of Technology (Japan), Electrotechnical Laboratory (Japan), ZIB/Freie Universität Berlin (Germany), University of Paraborn (Germany), University of Cardiff (UK), University of Lecce (Italy), CERN, (Switzerland), Porzan Supercomputing Centre (Poland), and CNUCE-Institute of the Italian National Research Council. We would like to thank our colleagues from these organizations for providing access to their supercomputing resources.

References

[1] Abramson, D., Giddy, J., and Kotler, L., *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?*, International Parallel and Distributed Processing Symposium (IPDPS 2000), Cancun, Mexico, May 2000,

[2] Baker M., Buyya R., Laforenza D., *The Grid: International Efforts in Global Computing*, Intl. Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR 2000), Italy, 2000.

[3] Buyya, R., Abramson, D., and Giddy, J., *Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid*, 4th International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2000), Beijing, China.

[4] Buyya, R., Abramson, D., and Giddy, J., *An Economy Driven Resource Management Architecture for Global Computational Power Grids*, The 7th International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA, June 26-29, 2000.

[5] Buyya, R., Abramson, D., and Giddy, J., *An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications*, The 2nd International Workshop on Active Middleware Services (AMS 2000), August 1, 2000, Pittsburgh, USA (Kluwer Academic Press).

[6] Buyya, R., Abramson, D., and Giddy, J., Stockinger H., *Economic Models for Resource Trading in a Service Oriented Grid Computing Environments*, Monash University, http://www.buyya.com/ecogrid/, Oct 2000 (in publication).

[7] J. Basney and M. Livny, "Deploying a High Throughput Computing Cluster", in High Performance Cluster Computing, R. Buyya, Editor, Vol. 1, Prentice Hall PTR, May 1999, http://www.cs.wisc.edu/condor/

[8] Czajkowski K., Foster I., Karonis N., Kesselman C., Martin S., Smith W., and Tuecke S., *A Resource Management Architecture for Metacomputing Systems*, IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[9] Distributed.Net – http://www.distributed.net/

[10] H.E. Bal et al.: "The distributed ASCI supercomputer project", ACM Special Interest Group, Operating Systems Review, Vol. 34, No. 4, October 2000. http://www.cs.vu.nl/das/

[11] European Grid (eGrid) and Application Testbeds – http://www.egrid.org, Nov. 2000.

[12] Foster I. and Kesselman C., *Globus: A Metacomputing Infrastructure Toolkit*, International Journal of Supercomputer Applications, 11(2): 115-128, 1997. (http://www.globus.org)

[13] Foster, I., and Kesselman, C. (editors), *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, USA, 1999.

[14] Globus Testbeds - http://www-fp.globus.org/testbeds/

[15] Global Grid Forum - http://www.gridforum.org/

[16] R. Buyya, Grid Computing Info Centre (Grid Infoware) – http://www.gridcomputing.com, 2000.

[17] Spyros Lalis and Alexandros Karipidis, *An Open Market-Based Framework for Distributed Computing over the Internet*, First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Dec. 2000, Bangalore, India: Springer Verlag, Germany.

[18] S. Chapin, J. Karpovich, A. Grimshaw, *The Legion Resource Management System*, Proceedings of the 5th Workshop on Job Scheduling Strategies for Parallel Processing, April 1999. (http://legion.virginia.edu/)

[19] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, C. Staelin, *An Economic Paradigm for Query Processing and Data Migration in Mariposa*, Proceedings of 3rd International Conference on Parallel and Distributed Information Systems, Austin, TX, USA, 28-30 Sept. 1994. Los Alamitos, CA, USA: IEEE Comput. Soc. Press, 1994.

[20] SETI@Home – http://setiathome.ssl.berkeley.edu/

[21] Gerald J. Tesauro and Jeffrey O. Kephart, *Foresight-based pricing algorithms in an economy of software agents*, First International Conference on Information and Computation Economies, Charleston, South Carolina, October, 1998. http://www.ibm.com/iac/papers/ice98_fs/

[22] Jakka Sairamesh and Jeffrey O. Kephart, *Price Dynamics of Vertically Differentiated Information Markets*, First

International Conference on Information and Computation Economies, Charleston, South Carolina, October, 1998.

[23] Manoj Kumar and Stuart I. Feldman, *Business negotiations on the Internet*, Technical Report, IBM Institute of Advanced Commerce - March 11, 1998, http://www.ibm.com/iac/tech-paper.html

[24] Foundation for Intelligent Physical Agents (FIPA), http://www.fipa.org/, 2000.

[25] Mojo Nation - http://www.mojonation.net/, Nov. 2000.

[26] Reid Smith and Randall Davis, *The Contract Net Protocol: High Level Communication and Control in a Distributed Problem Solver*, IEEE Transactions on Computers, Vol. C-29, No. 12, Dec. 1980.

[27] ISO New England Inc., Electricity Trading Over the Internet Begins in Six New England States, Holyoke, Massachusetts, http://www.iso-ne.com/ , Business Wire, http://industry.java.sun.com/javanews/stories/story2/0,1072,15093,00.html

[28] R. Cocchi, S. Shanker, D. Estrin, and L. Zhang, Pricing in Computer Networks: Motivation, Formulation, and Example, IEEE/ACM Transactions on Networking, Vol. 1, No. 6, Dec. 1993.

[29] Brent Chun and David Culler, *Market-based proportional resource sharing for clusters*, Technical report, University of California, Berkeley, September 1999.

[30] Heiser G, Lam F, and Russell SM, *Resource Management in the Mungi Single-Address-Space Operating System*, Proceedings of Australasian Computer Science Conference, Perth Australia, Feb. 1998, Springer-Verlag, Singapore, 1998.

[31] Yair Amir, Baruch Awerbuch and R. Sean Borgstrom, A Cost-Benefit Framework for Online Management of a Metacomputing System, Proceedings of first International Conference on Information and Computational Economy, Charleston, Oct. 1998.

[32] Amir Y., Awerbuch B., Barak A., Borgstrom R.S. and Keren A., An Opportunity Cost Approach for Job Assignment in a Scalable Computing Cluster, IEEE Tran. Parallel and Distributed Systems, Vol. 11, No. 7, July 2000.

[33] Noam Nisan, Shmulik London, Ori Regev, Noam Camiel, *Globally Distributed computation over the Internet - The POPCORN project*, International Conference on Distributed Computing Systems (ICDCS'98) 1998. Also a poster in WWW6 - Sixth International World Wide Web Conference, Santa-Clara in Aprill 1997. (http://www.cs.huji.ac.il/~popcorn/)

[34] Dickon Reed, Ian Pratt, Paul Menage, Stephen Early, Neil Stratford, *Xenoservers; Accounted execution of untrusted code*, Hot topics in Operating Systems, 1999. http://www.cl.cam.ac.uk/Research/SRG/netos/xeno/

[35] Jonathan Bredin and David Kotz and Daniela Rus, *Utility Driven Mobile-Agent Scheduling*, Technical Report PCS-TR98-331, Dept. of Computer Science, Dartmouth College, October 3, 1998.

[36] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta, Spawn: A Distributed Computational Economy, IEEE Transactions on Software Engineering, Vol. 18, No. 2, Feb. 1992.

[37] Scott Jackson, *QBank: A Resource Management Package for Parallel Computers*, Pacific Northwest National Laboratory, Washington, USA, 2000.

[38] B. Clifford Neuman and Gennady Medvinsky, *Requirements for Network Payment: The NetCheque Perspective*, Proceedings of IEEE COMPCON'95: Technologies for the Information Superhighway, San Francisco, USA, March 1995.

[39] Gennady Medvinsky and B. Clifford Neuman, *NetCash: A design for practical electronic currency on the Internet*. Proceedings of 1st the ACM Conference on Computer and Communication Security, November 1993.

[40] A. Lazar and N. Semret, *Auctions for Network Resource Sharing*, TR 468-97-02, Columbia University, Feb. 1997.

[41] Tuomas Sandholm, *Distributed Rational Decision Making*, Multiagent Systems (G. Weiss, editor), The MIT Press, 2000.

[42] Michael Huhns and Larry Stephens, *Mutiagent Systems and Societies of Agents*, Multiagent Systems (G. Weiss, editor), The MIT Press, 2000.

[43] John Brooke, Martyn Foster, Stephen Pickles, Keith Taylor, and Terry Hewitt, *Mini-Grids: Effective Test-beds for Grid Application*, Proceedings of the First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Dec. 17, 2000, Bangalore, India: Springer Verlag Press, Germany.

[44] Lee McKnight and Jahangir Boroumand, Pricing Internet Services: Approaches and Challenges, IEEE Computer, Feb. 2000.

[45] Lee McKnight and Jahangir Boroumand, Pricing Internet Services: After Flat Rate, MIT/Tufts Internet QoS Workshop, Dec. 1999.

[46] S. Smale, Dynamics in general equilibrium theory, American Economic Review, Vol. 66, No. 2, 284-294pp, May 1976.

[47] Paypal – http://www.paypall.com, 2000.

[48] Bob Knighten, *Peer to Peer Computing* Working Group, Intel Developer's Forum, August 24, 2000, http://www.peer-to-peerwg.org

[49] Napster - http://www.napster.com/

[50] Gnutella - http://gnutella.wego.com/

[51] M. Miller and K. Drexler, *Markets and Computation: Agoric Open Systems*, The Ecology of Computation, B. Huberman (editor), Elsevier Science Publishers, The Netherlands, 1998.

SYSTEM NAME	ECONOMY MODEL	PLATFORM	COMMENTS
Mariposa [19] (UC Berkeley)	Bidding (Tendering/ ContractNet). Pricing based on load and historical info.	Distributed database.	It supports budget-based query processing and storage management.
Mungi [30] (UNSW, Sydney) (It is a single-address-space operating system)	Commodity market (renting storage space that increases as available storage runs low, forcing users to release unneeded storage.)	Storage servers.	It supports storage objects based on bank accounts from which rent is collected for the storage occupied by objects. .
Popcorn [33] (Hebrew Uni., Israel)	Auction. (Highest bidder gets access to resource and it transfers credits from buyer to the seller account.)	Web browsers. (*Popcorn based parallel code* run within a browser of CPU cycles seller.)	Popcorn API-based parallel applications need to specify a budget for processing each of its modules.
Java Market [31] (John Hopkins Uni)	QoS based computational market. (The resource owner receives f(j, t) award for completing f in time t.)	Web browsers. (JavaMarket runs *standard Java Applets* within a browser).	One can sell CPU cycles by pointing Java-enabled browser to Portal & allow execution of Applets.
Enhanced MOSIX [32] (Hebrew Uni., Israel)	Commodity market (resource cost of each node is known)	Clusters of computers (Linux PCs)	It supports process migration such that overall cost of job execution is kept low.
JaWS [17] (Uni. of Crete, Greece)	Bidding (Tendering)	Web browsers	It is similar to Popcorn.
Xenoservers [34] (University of Cambridge)	Bidding (Proportional resource sharing)	Single computer	Accounted execution of untrusted code.
D'Agents [35] (Dartmouth College)	Bidding (Proportional resource sharing)	Single computer or Mobile Agents	Agents bid function is proportional to benefit.
Rexec/Anemone [29] (UC Berkeley)	Bidding/Auction (for proportional resource sharing)	Clusters (A market-based Cluster Batch Queue System)	Users assign utility value to their application and system allocates resources proportionally.
Mojo Nation [25] (Autonomous Zone Industries, CA)	A Credit-based partnership and/or bartering model. (Contributors earn credits by sharing storage and spend them when required)	Network storage.	It is a content-sharing community network. It combines marketplace and bartering approach for file/resource sharing.
Spawn [36] (Xerox PARC)	Second-price Auction (uses sponsorship model for funding money to each task depending on some requirements)	Network on workstations. Each WS executes a single task per time slice	It supports execution of concurrent program expressed in the form of hierarchy of processes that expand and shrink size depending on the resource cost.
Supercomputing centers [43] (e.g., Uni. of Manchester computing services for academic research)	Commodity market and priority-based model (they charge for CPU, memory, storage, and human support services)	MPPs, Crays, and Clusters, and Storage servers.	Any application can use this service and QoS is proportional to user priority and scheduling mechanisms.

Table 1: A computational economy based resource management systems.

Resource Type & Size (No. of nodes)	Organization & Location	Grid Services and Fabric	Price @ AU peak time	Price @ AU off peak time.
Linux cluster (60 nodes)	Monash, Australia	Globus/Condor	20	5
IBM SP2 (80 nodes)	ANL, Chicago, USA	Globus/LL	5	10
Sun (8 nodes)	ANL, Chicago, USA	Globus/Fork	5	10
SGI (96 nodes)	ANL, Chicago, USA	Globus/Condor-G	15	15
SGI (10 nodes)	ISI, Los Angeles, USA	Globus/Fork	10	20

Table 2: A Sample of Economy Grid Testbed Resources used in the experiment. Price is G$ per CPU sec.

Rajkumar Buyya is a Doctoral Candidate at the School of Computer Science and Software Engineering, Monash University, Melbourne, Australia. He was awarded Dharma Ratnakara Memorial Trust Gold Medal for his academic excellence during 1992 by Mysore and Kuvempu Universities. He has authored three books *Microprocessor x86 Programming*, *Mastering C++*, and *Design of PARAS Microkernel*. He has edited a popular two volumes book on *High Performance Cluster Computing* published by Prentice Hall, USA. He also edited proceedings of six international conferences and served as guest editor for major research journals. He has contributed to the development of system software for PARAM supercomputers produced by the Centre for Development of Advanced Computing (C-DAC), India. At Monash University, he is conducting R&D on next generation Internet/Grid computing technologies and its applications. Rajkumar is a speaker in the IEEE Computer Society Chapter Tutorials Program and Co-founder/Chair of the IEEE Computer Society Task Force on Cluster Computing (TFCC). He has organised and chaired IEEE/ACM international conferences in the area of Cluster and Grid Computing. He has lectured on advanced technologies such as Parallel, Distributed and Multithreaded Computing, Internet and Java, Cluster Computing, and Java for High Performance Computing in many international conferences and institutions.

David Abramson is a professor and the Head of the School of Computer Science and Software Engineering (CSSE) at Monash University, Melbourne, Australia. He has been involved in computer architecture and high performance computing research since 1979. Previous to joining Monash University in 1997, he has held appointments at Griffith University, CSIRO, and RMIT. At CSIRO he was the program leader of the Division of Information Technology High Performance Computing Program, and was also an adjunct Associate Professor at RMIT in Melbourne. He was also a program manager in the Co-operative Research Centre for Intelligent Decisions Systems. Abramson is currently project leader in the Co-operative Research Centre for Distributed Systems Nimrod Project. He is also Chief Investigator in the following ARC funded research projects: Guard - A Relative Debugger and a Software Environment for Building High Performance Optimising Decision Support Systems from Computational Models. Abramson has chaired a number of international conferences, including the prestigious ACM International Symposium on Computer Architecture in 1992. He has published over 100 papers and technical documents. He has given seminars and received awards around Australia and internationally and has received over $1 million in research grants. He is a co-founder of Active Tools P/L, a company that was established to commercialise the Nimrod project.

Jon Giddy is a Research Scientist at the Distributed Systems Technology Centre (DSTC), Monash University, Melbourne, Australia. He holds BSc honours in Computer Science from the University of Wollongong. He worked on a number of DSTC funded projects—Security Unit at the Queensland University of Technology and Tools Unit at the Griffith University. In the context of Nimrod project, he is involved in design and development of number of software tools for high performance distributed computing. He enjoys programming in Python!