The Grid Economy

RAJKUMAR BUYYA, DAVID ABRAMSON, AND SRIKUMAR VENUGOPAL

Invited Paper

This paper identifies challenges in managing resources in a Grid computing environment and proposes computational economy as a metaphor for effective management of resources and application scheduling. It identifies distributed resource management challenges and requirements of economy-based Grid systems, and discusses various representative economy-based systems, both historical and emerging, for cooperative and competitive trading of resources such as CPU cycles, storage, and network bandwidth. It presents an extensible, service-oriented Grid architecture driven by Grid economy and an approach for its realization by leveraging various existing Grid technologies. It also presents commodity and auction models for resource allocation. The use of commodity economy model for resource management and application scheduling in both computational and data grids is also presented.

Keywords*—Distributed computing, grid economy, resource management, utility computing.*

I. INTRODUCTION

Inspired by the electrical power Grid's pervasiveness, ease of use, and reliability, computer scientists in the mid-1990s began exploring the design and development of an analogous infrastructure called the *computational power Grid* [23] for wide-area parallel and distributed computing. The motivation for computational Grids was initially driven by large-scale, resource (computational and data) intensive scientific applications that require more resource than a single computer (PC, workstation, supercomputer, or cluster) could provide in a single administrative domain. A Grid enables the sharing, selection, and aggregation of a wide variety of geographically distributed resources including supercomputers, storage systems, data sources, and specialized devices owned by different organizations for solving large-scale resource intensive problems in science, engineering, and commerce.

Manuscript received March 1, 2004; revised June 1, 2004.

R. Buyya is with the Grid Computing and Distributed Systems Laboratory and NICTA Victoria Laboratory, Department of Computer Science and Software Engineering, University of Melbourne, VIC 3010, Australia (e-mail: raj@csse.unimelb.edu.au).

D. Abramson is with the School of Computer Science and Software Engineering, Monash University, VIC 3163, Australia.

S. Venugopal is with the Grid Computing and Distributed Systems Laboratory, Department of Computer Science and Software Engineering, University of Melbourne, VIC 3010, Australia.

Digital Object Identifier 10.1109/JPROC.2004.842784

To build a Grid, the development and deployment of a number of services is required. They include *low-level services* such as security, information, directory, and resource management [resource trading, resource allocation, and quality-of-services (QoSs)] and *high-level services/tools* for application development, resource management, and scheduling (resource discovery, access cost negotiation, resource selection, scheduling strategies, QoSs, and execution management) [23], [35], [52], [50], [51]. Among them, the two most challenging aspects of Grid computing are resource management and scheduling.

A. Resource Management and Scheduling

In Grid environments, the *producers* (resource owners) and *consumers* (resource users) have different goals, objectives, strategies, and supply-and-demand patterns. More importantly both resources and end users are geographically distributed in different time zones. The most commonly used approaches for managing such complex environments are driven by s *ystem-centric* and *user-centric* policies. System-centric is a traditional approach to resource management that attempts to optimize system-wide measure of performance and is commonly used in managing resources in single administrative domains. User-centric approaches, on the other hand, concentrate on delivering maximum utility to the users of the system based on their QoS requirements, i.e., a guarantee of certain levels of performance based on the attributes that the user finds important such as the deadline by which his jobs have to be completed. Enforcing QoS requires a system of rewards and penalties and, hence, it is common to find user-centric approaches driven by economic principles.

System-centric Grid resource management systems such as Legion [64], Condor [41], AppLeS PST [15], [21], NetSolve [20], PUNCH [47], and XtremWeb [17] adopt a *conventional strategy*, where a scheduling component decides which jobs are to be executed at which resource based on cost functions driven by system-centric parameters. They aim to enhance the system throughput, utilization, and complete execution at the earliest possible time rather than improving the utility of application processing. They do not take resource access cost (price) into consideration, which means that the value of processing applications at any time is treated the

same, which is not the case in reality—the value should be higher when there is a production schedule deadline. The end user does not want to pay the highest price but wants to negotiate a particular price based on the demand, value, priority, and available budget.

In an economy-based approach, scheduling decisions are made dynamically at runtime and are driven and directed by the end-users requirements. While a conventional cost model often deals with software and hardware costs for running applications, an economy model primarily charges the end user for services that they consume based on the value they derive from it. Pricing policies are based on the demand from the users and the supply of resources is the main driver in the competitive, economic market model. Therefore, a user competes with other users and a resource owner with other resource owners.

Traditional approaches use centralized policies that need complete state information and a common fabric management policy, or a decentralized consensus based policy. The economic approach provides a fair basis in successfully managing the decentralization and heterogeneity that is present in human economies. Competitive economic models provide algorithms/policies and tools for resource sharing or allocation in Grid systems. These models can be based on bartering or prices. In the *bartering-based model*, all participants need to own resources and trade resources by exchanges (e.g., storage space for CPU time). In the *price-based model*, the resources have a price, based on the demand, supply, value, and the wealth in the economic system. In addition, it enhances the social structure of the Grid thereby ensuring its stability and efficiency [16].

The resource management and scheduling systems for Grid computing need to manage resources and application execution depending on resource consumers' and owners' requirements, and they need to continuously adapt to changes in the availability of resources. This requirement introduces a number of challenging issues that need to be addressed such as, site autonomy, heterogeneous substrate, policy extensibility, resource allocation or co-allocation, online control, resource trading, and QoS-based scheduling. A number of Grid systems (such as Globus [24] and [32]) have addressed many of these issues with the exception of *resource trading* and *quality of service-based scheduling*. The Grid Economy framework presented in this chapter addresses these two issues. It leverages existing middleware technologies and provides new services that are essential for resource trading and aggregation, depending on their availability, capability, cost, and users' QoS requirements.

B. Computational Economy: Assessing Wants and Needs

In an economic-based Grid computing environment, resource management systems need to provide mechanisms and tools that allow resource consumers (end users) and providers (resource owners) to express their requirements and facilitate the realization of their goals. Resource consumers need a utility model—how consumers demand resources and their preference parameters, and a broker that supports resource discovery and strategies for dynamically scheduling applications on distributed resources at runtime depending on their availability, capability, and cost along with user-defined QoS requirements. The resource providers need tools and mechanisms that support price specification and generation schemes to increase system utilization, and protocols that support service publication, trading, and accounting. For the market to be competitive and healthy, coordination mechanisms are required so that equilibrium is achieved, i.e., the supply of a service equals the quantity demanded.

Numerous economic models including microeconomic and macroeconomic principles for resource management have been proposed in the literature [3], [26], [40], [43], [51], [56], [61]. These include: commodity market models, posted price models, bargaining models, tendering, or contract-net models, auction models, bid-based proportional resource sharing models, cooperative bartering models, and monopoly and oligopoly.

In general, the benefits of Grid economies can be listed as follows.

- It helps in building a large-scale Grid as it offers incentive for resource owners to contribute their resources for others to use and profit from it.
- It helps in regulating the supply and demand for resources.
- It offers an economic incentive for users to reduce their priority in favor of incurring a lesser expense and, thus, encourages the solution of time critical problems first.
- It provides a common basis for comparing conflicting needs by allowing users to express their requirements and objectives in currency terms.
- It offers uniform treatment of all resources. That is, it allows trading of everything including computational power, memory, storage, network bandwidth/latency [58], data, and devices or instruments.
- It helps in building a highly scalable system as the decision-making process is distributed across all users and resource owners.
- It supports a simple and effective basis for offering differentiated services for different applications at different times.

Finally, it places the power in the hands of both resource owners and users—they can make their own decisions to maximize the utility gained and profit.

C. Requirements for Economic-Based Grid Systems

Economic-based resource management systems need to provide mechanisms and tools that allow resource consumers (end users) and providers (resource owners) to express their requirements and facilitate decision-making to further their objectives [9]. That is, they need 1) the means to express their valuations and objectives [*value expression*], 2) scheduling policies to translate them to resource allocations [*value translation*], and 3) mechanisms to enforce selection and allocation of differential services, and dynamic adaptation to changes in their availability at runtime [*value enforcement*]. Similar requirements are raised [7] for market-based systems in a single administrative domain environment such as clusters. However, they are limited to co-operative economic models since they aim for social welfare. Grids need to use

competitive economic models as resource providers and resource consumers have varying goals, strategies, and requirements that vary with time.

Essentially, resource consumers need a *utility model*—to allow them to specify resource requirements and constraints. They need *brokers* that provide strategies for choosing appropriate resources [*value translation*] and dynamically adapt to changes in resource availability at runtime to meet user requirements [*value enforcement*]. The resource owners need mechanisms for *price generation schemes* to increase system utilization and *protocols* that help them offer competitive services [*value expression*]. Grid resources have their schedulers (e.g., OS or queuing system) that allocate resources [*value translation*]. Some research systems support resource reservation in advance (e.g., reserving a slot from time t1 to t2 using the Globus GARA [25] and bind a job to it) and allocate resources during reserved time [*value enforcement*]. A number of research systems have explored QoS based resource (e.g., CPU time and network bandwidth [58], [3]) allocation in operating systems and queuing systems, but the inclusion of QoS into mainstream systems has been slow paced (e.g., the Internet mostly uses the best effort allocation policy [35], but this is changing with IPv6 [5]).

An economic approach to Grid computing introduces a number of new issues like resource trading and QoS-based scheduling in addition to those such as site autonomy, heterogeneous substrate, policy extensibility, online control already addressed by existing Grid systems. To address these new issues, the economy-based Grid systems need to support 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.
- Users' QoS requirements-driven brokering/scheduling systems.

II. Representative Works

Various criteria used for judging effectiveness of a market model are [69]: social welfare (global good of all), Pareto efficiency (global perspective), individual rationality (better off by participating in negotiation), stability (mechanisms that cannot be manipulated, i.e., behave in the desired manner), computational efficiency (protocols should not consume too much computation time), and distribution and communication efficiency (communication overhead to capture a desirable global solution).

Several research systems (see Table 1) have explored the use of different economic models for trading resources to manage resources in different application domains: CPU cycles, storage space, database query processing, and distributed computing, They include Spawn [10], Popcorn [48], Java Market [72], Enhanced MOSIX [73], JaWS [66], Xenoservers [14], D'Agents [28], Rexec/Anemone [6], Mojo Nation [45], Mariposa [44], Mungi [18], Stanford Peers [8], G-Commerce [62], OCEAN [39], Nimrod-G [52], and GridSim [49], and Gridbus [57].

Each of the resource management systems presented 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 using monolithic system architecture that limits their extensibility.

III. Grid Architecture for Computational Economy

A distributed grid architecture for computational economy (GRACE) is shown in Fig. 1 and has been enhanced to support computational, data, and service-oriented Grids. This architecture is generic enough to accommodate different economic models used for resource trading. The key components of the Grid include the following.

- Grid User with Applications (sequential,arametric, parallel, or collaborative applications).
- Programming Environments.
- User-Level Middleware and Tools such as GRBs.
- Core Grid Middleware (services for resource trading and coupling distributed wide area resources).
- Grid Service Providers (GSPs).

GRACE provides services that help both resource owners and users maximize their objective functions. The resource providers can contribute their resources 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. The resource brokers work for the consumers and attempt to maximize user utility. They can use GRACE services for resource trading and identifying GSPs that meets its requirements.

Both GRBs and GSPs can initiate resource trading and participate in the interaction depending on their requirements and objectives. GRBs may invite bids from a number of GSPs and select those that offer the lowest service costs and meet their deadline and budget requirements. Alternatively, GSPs may invite bids in an auction and offer services to the highest bidder as long as its objectives are met. Both GSPs and GRBs have their own utility functions that must be satisfied and maximized. The GRBs perform a cost-benefit analysis depending on the deadline (by which the results are required) and budget available (the amount of money the user is willing to invest for solving the problem). The resource owners decide their pricing based on various factors. They may charge different prices for different users for the same service or it can vary depending on the specific user demands. Resources may have different prices based on environmental influences such as the availability of larger core memory and better communication bandwidth with the outside world.

Grid brokers (note that in a Grid environment each user has his/her own broker as his agent) may have different goals

Table 1 Computational Economy Based Distributed Resource Management Systems

System Name	Economic Model	Platform	Remarks
Mariposa [44] (UC Berkeley)	Bidding (Tendering/ ContractNet). Pricing based on load and historical info.	Distributed database.	It supports budget-based query processing and storage management.
Mungi [18] (University of New South Wales)	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 [48] (Hebrew University)	Auction. (Highest bidder gets access to resource and it transfers credits from buyer to the seller account.)	Web browsers. (*Popcorn parallel code* runs 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 [72] (Johns Hopkins University)	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 [73] (Hebrew U., 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 [66] (University of Crete)	Bidding (Tendering)	Web browsers	It is similar to Popcorn.
Xenoservers [14] (University of Cambridge)	Bidding (Proportional resource sharing)	Single computer	Accounted execution of un-trusted code.
D'Agents [28] (Dartmouth College)	Bidding (Proportional resource sharing)	Single computer or Mobile Agents	Agents bid function is proportional to benefit.
Rexec/Anemone [6] (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 [45] (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 [10] (Xerox PARC)	Second-price/Vickery 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.
CSAR Supercomputing center [29] (University of Manchester)	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.
Nimrod-G [52] (Monash University)	It supports economy models such as commodity market, spot market, and contract-net for price establishment.	World Wide Grid (resources Grid enabled using Globus middleware)	It is a *real* system that supports deadline and budget constrained algorithms for scheduling task-farming and data parallel applications on world-wide distributed resources depending on their cost, power, availability and users quality of service requirements.
GridSim [49] (Monash University)	Currently, it supports economic models similar to those used in Nimrod-G, but limited to them.	A Java-based discrete event toolkit for simulating Grid resources, users, applications, and brokers.	The economic Grid resource broker supports deadline and budget based time, cost, cost-time, and conservative time optimisation scheduling algorithms.
G-Commerce [62] (U. of California Santa Barbara)	Commodity and auctions	Simulates hypothetical consumers and produces.	It is exploring strategies for pricing Grid resources to enable resource trading.
OCEAN [39] (U. of Florida)	Continuous double auction	A Java based platform with distributed PCs.	It is exploring the use of continuous double auction for trading computational resources – in development.
Stanford Peers [8] (Stanford University)	Auctions with cooperative bartering in a cooperative sharing environment.	Simulates storage trading for content replication and archiving.	It demonstrates distributed resource trading policies based on auctions by *simulation* – in development.
Gridbus [57] (U. of Melbourne)	Both commodity and auction models for clusters, computational and data grids	Clusters and the World-Wide Grid	It provides services for ensuring end-to-end Quality of Services at various levels for users in cluster and Grid environments.

(e.g., different deadlines and budgets), and each broker tries to maximize its own good without concern for the global good. This needs to be taken into consideration in building automated negotiation infrastructure. In a *cooperative distributed computing or problem-solving environment* (like cluster computers or a federation of clusters), the system

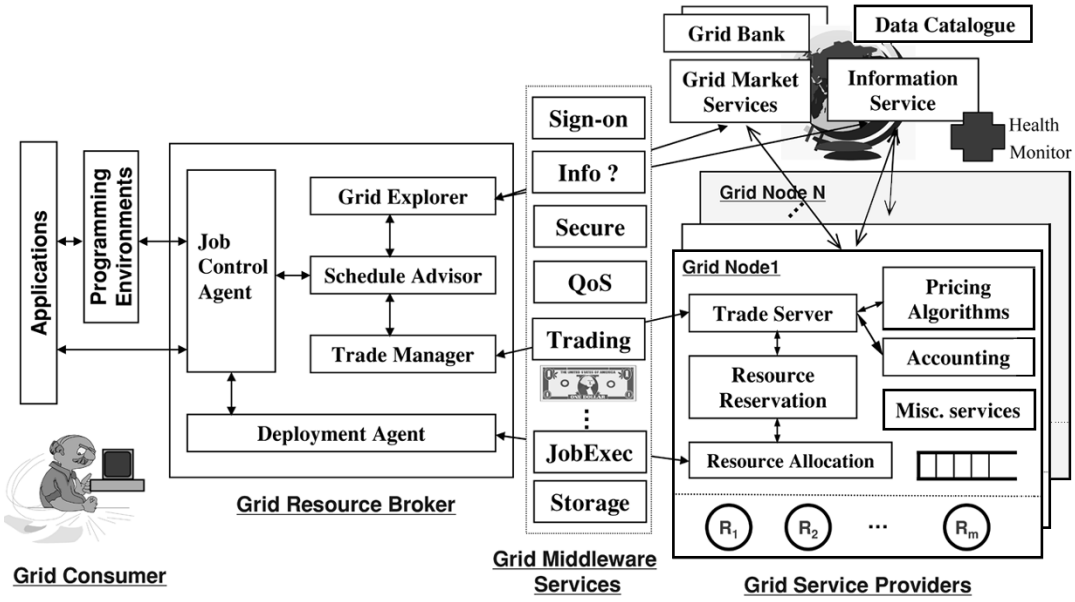


Fig. 1. A Generic Grid architecture for computational economy.

designers impose an *interaction protocol* (possible actions to take at different points) and a *strategy* (a mapping from one state to another and a way to use the protocol). This model aims for global efficiency as nodes cooperate toward a common goal. On the other hand, in Grid systems, brokers, and GSPs are provided with an interaction protocol, but they choose their own private strategy (similar to multi-agent systems), which cannot be imposed from outside. Therefore, the negotiation protocols need to be designed assuming a *noncooperative, strategic* perspective. In this case, the main concern is the social outcomes that follow given a protocol and that guarantees that each broker/GSPs desired local strategy is best for that broker/GSP and hence the broker/GSP will use it.

A. 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 resource broker consists of the following components.

- *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 the Grid explorer), resource selection and job assignment (schedule generation) to ensure that the user requirements are met.
- *Grid Explorer (GE):* This is responsible for resource discovery by interacting with the 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 (the schedule advisor) to identify resource access costs. It uses market directory services and GRACE negotiation services for trading with Grid service providers (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 updates the status of task execution to JCA.

B. Core Middleware Level and Grid Economy

Traditionally core Grid middleware focused on providing services required for secure and uniform access to distributed resources. They include security, single sign-on, remote process management, storage access, data management, and information services. These services are being standardized via efforts such as Web Services Resource Framework (WSRF) [33] to support the use of service-oriented architectures in distributed systems and applications development. Core middleware technologies such as Globus are aiming at providing standards-based software services. In the Grid economy context, GSPs specifically need to deal with the following components that can be considered as part of the services-driven next-generation core grid middleware.

- *Grid Market Directory (GMD):* It allows resource owners to publish their services in order to attract consumers.
- *Grid Trade Server (GTS):* 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. It consults pricing policies during negotiation and directs the accounting system to record resource consumption and to bill the user according to the agreed pricing policy.
- *Pricing Policies:* These define the prices that resource owners would like to charge users. The resource

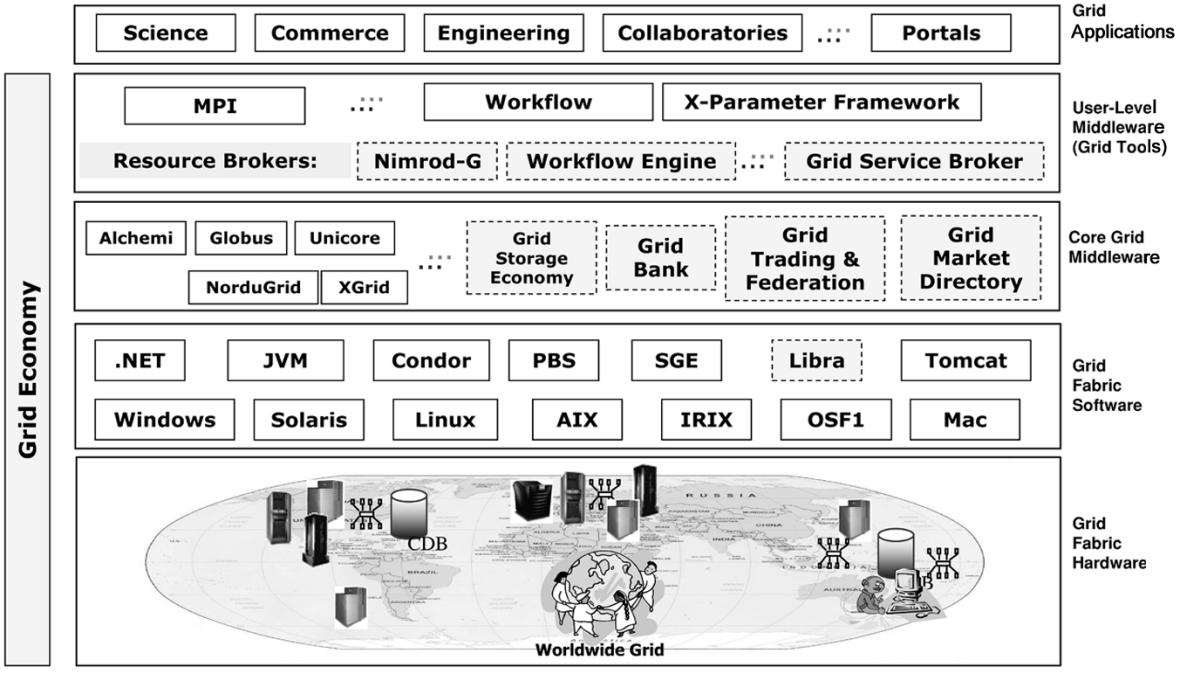


Fig. 2. Scenario for realizing the GRACE framework.

owners may follow various policies to maximize their profit and resource utilization and the price they charge may vary with time and user population. Also, the pricing can be driven by demand and supply as in the real market environment. That is, within commodity market model, pricing is essentially determined by objective functions of service providers and users. The pricing policy can also be based on auction. Within auction based economic model, pricing is driven by how much value users place on the service and access to Grid services is won by the bidder whose valuation comes closest to that of the resource owner.

- *Resource Accounting and Charging:* components such as Grid Bank [1] along with QBank [65] are responsible for recording resource usage and bill the user as per the usage agreement between resource broker (a user agent) and trade server (resource owner agent).

The service providers publish their services through the GMD. They use Grid trading services' declarative language for defining cost specification and their objectives such as access price for various users for different times and durations, along with possibilities of offering discounts to attract users during off-peak hours. The GTS can employ different economic models in providing services. The simplest would be a commodity model wherein the resource owners define pricing strategies including those driven by the demand and resource availability. The GTS can act as auctioneer if the Auction-based model is used in deciding the service access price or an external auctioneer service can be used.

A layered architecture for the realization of the GRACE framework is shown in Fig. 2. It offers Grid economy infrastructure that co-exists with or is built on top of the existing middleware such as Globus, Alchemi [4] and Unicore [27]. The impact of Grid economy can be felt at various levels of system architecture: local resource management, resource access mediator services provided by core middleware, resource brokers while making selection of resources, a programming framework and policy that allocates budget for different activities of an application.

- Problem solving environments with built in schedulers (e.g., ActiveSheets [12] on Nimrod-G [11]).
- Programming frameworks and development tools (e.g., Nimrod parameter specification language [13]).
- A resource broker (e.g., Nimrod-G).
- Various resource trading protocols.
- A mediator for negotiating between users and Grid service providers (Grid Market Directory [31]).
- A deal template for specifying resource requirements and services offers.
- A trade server.
- A pricing policy specification.
- Accounting (e.g., QBank [65]) and payment management (GridBank [1]).
- A local resource management system allocating resources based on service-level agreements (e.g., Libra [30]).

The new middleware services being proposed are designed to offer low-level services that co-exist with existing low-level middleware services and infrastructure. Higher-level services and tools such as the Nimrod-G Resource Broker, which uses economic models suitable for meeting the user requirements, can use these core services.

IV. Economy Within Computational Grids

The integration of computational economy as part of a scheduling system greatly influences the way computational resources are selected to meet the user requirements. The users should be able to submit their application along with their requirements to a scheduling system such as Nimrod-G,

which can process the application on the Grid on the user's behalf and try to complete the assigned work within a given deadline and cost. The deadline represents a time by which the user requires the result, and is often imposed by external factors like production schedules or research deadlines.

To arrive at a scheduling decision, the scheduling system needs to take various parameters into consideration including resource architecture and configuration, resource state (such as CPU load, memory available, disk storage free), resource requirements of an application, resource availability, network bandwidth, Load, and Latency, historical Information such job consumption rate. The important parameters of computational economy that can influence the way resource scheduling is done are the following.

- Resource Cost (set by its owner).
- Price (that the user is willing to pay).
- Deadline (the period by which an application execution needs to be completed).

The scheduler can use the information gathered by a resource discoverer and also negotiate with resource owners to establish service price. The resource that offers the best price and meets resource requirements can eventually be selected. This can be achieved by resource reservation and bidding. If the user deadline is relaxed, the chances of obtaining low-cost access to resources are high. The cost of resources can vary with time and the resource owner will have the full control over deciding access cost. Further, the cost can vary from one user to another. The scheduler can even solicit bids from resource providers in an open market, and select the feasible service-provider(s). To accomplish this, we need scheduling algorithms that take the application processing requirements, Grid resource dynamics, and the user QoS requirements such as the deadline, budget, and their optimization preference into consideration. In this section, we discuss deadline and budget constrained (DBC) algorithms that we developed for scheduling parameter sweep applications on globally distributed Grid resources.

A. Scheduling Algorithms

The parameter sweep applications, created using a combination of task and data parallel models, contain a large number of independent jobs operating different data sets. A range of scenarios and parameters to be explored are applied to the program input values to generate different data sets. The programming and execution model of such applications resemble the SPMD (Single Program Multiple Data) model. The execution model essentially involves processing N independent jobs (each with the same task specification, but a different dataset) on M distributed computers where N is, typically, much larger than M.

The scheduling and orchestration of the execution of parameter sweep applications on world-wide distributed computers appears simple, but complexity arises when users place QoS constraints like deadline (execution completion time) and computation cost (budget) limitations. Such a guarantee of service is hard to provide in a Grid environment since its resources are shared, heterogeneous, distributed in nature, and owned by different organizations having their own policies and charging mechanisms. In addition, scheduling algorithms need to adapt to the changing load and resource availability conditions in the Grid in order to achieve performance and at the same time meet the deadline and budget constraints. In our Nimrod-G application level resource broker (also called an application level scheduler) for the Grid, we have incorporated four adaptive algorithms for deadline and budget constrained scheduling [56].

- Cost Optimization, within time and budget constraints.
- Time Optimization, within time and budget constraints.
- Conservative Time Optimization, within time and budget constraints.
- Cost-Time Optimization, within time and budget constraints.

Table 2 Deadline and Budget Constrained Scheduling Algorithms and Objectives

Scheduling Algorithm Strategies	*Execution Time (not beyond the deadline)*	*Execution Cost (not beyond the budget)*
Cost Optimisation	Limited by deadline	Minimise
Time Optimisation	Minimise	Limited by budget
Conservative Time Optimisation	Limited by deadline	Limited by budget
Cost-Time Optimisation	Limited by deadline	Limited by budget

The role of deadline and budget constraints in scheduling and objectives of different scheduling algorithms are illustrated in Table 2. In this following section we present two cost and time optimization-based scheduling algorithms and their performance results.

The *Time Optimization scheduling* algorithm attempts to complete the experiment as quickly as possible, within the budget available. A description of the core of the algorithm is as follows.

1) For each resource, calculate the next completion time for an assigned job, taking into account previously assigned jobs and job consumption rate.
2) Sort resources by next completion time.
3) Assign one job to the first resource for which the cost per job is less than or equal to the remaining budget per job.
4) Repeat the above steps until all jobs are assigned.

The *Cost Optimization scheduling* algorithm attempts to complete the experiment as economically as possible within the deadline.

1) Sort resources by increasing cost.
2) For each resource in order, assign as many jobs as possible to the resource, without exceeding the deadline.
3) Note that the implementations of all the above algorithms contain extra steps for dealing with the initial start-up (when the average completion times are unknown), and for when all jobs cannot be assigned to resources (infeasible schedules).

B. Scheduling Experiments

We have performed a number of deadline and budget constrained scheduling experiments with different requirements at different times by selecting different sets of resources available in the World Wide Grid (WWG) [55] testbed

```
#Parameters Declaration
parameter angle_degree integer range from 1 to 165 step 1;
parameter time_base_value integer default 5;

#Task Definition
task main
   #Copy necessary executables depending on node type
   copy calc.$OS node:calc
   #Execute program with parameter values on remote node
   node:execute ./calc $angle_degree $time_base_value
   #Copy results file to use home node with jobname as extension
   copy node:output ./output.$jobname
endtask
```

Fig. 3. Nimrod-G parameter sweep processing specification.

Table 3 The WWG Tested Resources Used in Scheduling Experiments, Job Execution and Costing

Resource Type & Size (No. of nodes)	Organization & Location	Grid Services and Fabric	Price (G$ per CPU sec.)	Jobs Executed on Resources	
				Time_Opt	Cost_Opt
Linux cluster (60 nodes)	Monash, Australia	Globus, GTS, Condor	2	64	153
Solaris (Ultra-2)	Tokyo Institute of Technology, Japan.	Globus, GTS, Fork	3	9	1
Linux PC (Prosecco)	CNUCE, Pisa, Italy	Globus, GTS, Fork	3	7	1
Linux PC (Barbera)	CNUCE, Pisa, Italy	Globus, GTS, Fork	4	6	1
Sun (8 nodes)	ANL, Chicago, USA	Globus, GTS, Fork	7	42	4
SGI (10 nodes)	ISI, Los Angeles, USA	Globus, GTS, Fork	8	37	5
			Total Experiment Cost (G$)	237000	115200
			Time to Complete Experiment (Min.)	70	119

during each experiment. They can be categorised into the following scenarios:

- cost optimization scheduling during australian peak and off-peak times,
- cost and time optimization scheduling using cheap local and expensive remote resources;
- large scale scheduling using cost and time optimization algorithms.

We briefly discuss the WWG testbed followed by a detailed discussion on these scheduling experiments.

1) The World-Wide Grid (WWG) Testbed: To enable our empirical research and experimentations in distributed computational economy and Grid computing, we created and expanded a testbed called the World-Wide Grid (WWG) in collaboration with colleagues from numerous organizations around the globe. The contributing organizations and the WWG resources are located in five continents: Asia, Australia, Europe, North America, and South America.

The WWG testbed contains numerous computers with different architecture, capability, and configuration. They include PCs, workstations, SMPs, clusters, and vector supercomputers running operating systems such as Linux, Sun Solaris, IBM AIX, SGI IRIX, and Compaq Tru64. Further, the systems use a variety of job management systems such as OS-Fork, NQS, Condor, RMS, PBS, and LSF. These system characteristics can be identified by accessing the GIS (Grid Information Service) provided by middleware systems such as Globus running on each resource.

2) Parameter Sweep Application: We have created a hypothetical parameter sweep application (PSA) that executes a CPU intensive program with 165 different parameter scenarios or values. The program *calc* takes two input parameters and saves results into a file named "output." The first input parameter *angle_degree* represents the value of angle in degree for processing trigonometric functions. The program *calc* needs to be explored for angular values from 1 to 165 degrees. The second parameter time_base_value indicates the expected calculation complexity in minutes plus 0–60 s positive deviation. That means the program *calc* is expected to run for anywhere between 5 and 6 min on resources with some variation depending on resource capability. A plan

file modeling this application as a parameter sweep application using the Nimrod-G parameter specification language is shown in Fig. 3. The first part defines parameters and the second part defines the task that needs to be performed for each job. As the parameter *angle_degree* is defined as a range parameter type with values varying from 1 to 165 in step of 1, it leads to the creation of 165 jobs with 165 different input parameter values. To execute each job on a Grid resource, the Nimrod-G resource broker, depending on its scheduling strategy, first copies the program executable(s) and necessary data to a Grid node, then executes the program, and finally copies results back to the user home node and stores output with job number as file extension.

3) Cost and Time Optimization Scheduling Using Local and Remote Resources: This experiment demonstrates the use of cheap local resources and expensive remote resources together for processing a parameter sweep application (same as used in the previous scheduling experiment) containing 165 CPU-intensive jobs, each running approximately 5-min duration. We have set the deadline of 2 h (120 min) and budget of 396 000 (G$ or tokens) and conducted experiments for two different optimization strategies.

- *Optimize for Time*—this strategy produces results as early as possible, but before a deadline and within a budget limit.
- *Optimize for Cost*—this strategy produces results by deadline, but reduces cost within a budget limit.

In these scheduling experiments, the Nimrod-G resource broker employed the *commodity market* model for establishing a service access price. The broker established connection with the Grid Trader running on resource providers' machines to obtain service prices at runtime. The broker architecture is generic enough to use any of the protocols discussed in [54] for negotiating access to resources and choosing appropriate ones. The access price varies for local and remote users: users are encouraged to use local resources since they are available at cheaper price. Depending on the deadline and the specified budget, the broker develops a plan for assigning jobs to resources. While doing so it does dynamic load profiling to establish the user job consumption rate for each resource. The broker uses this information to adapt itself to the changing resource conditions including failure of resources or jobs on the resource.

We have used a subset of resources of the WWG testbed in these scheduling experiments. Table 3 shows resources details such as architecture, location, and access price along with type of Grid middleware systems used in making them Grid enabled. These are shared resources and hence they were not fully available to us. The access price indicated in the table is being established dynamically using the GRACE resource trading protocols (commodity market model). The access price are artificial, however, they assigned to reflect the offering of differentiated services at different costs as in the real-world marketplace.

The number of jobs in execution on resources (Y-axis) at different times (X-axis) during the experimentation is shown in Fig. 4(a) and 4(b) for the time and cost optimization scheduling strategies, respectively. In the first (time minimization) experiment, the broker selected resources in such a way that the whole application execution is completed at the earliest

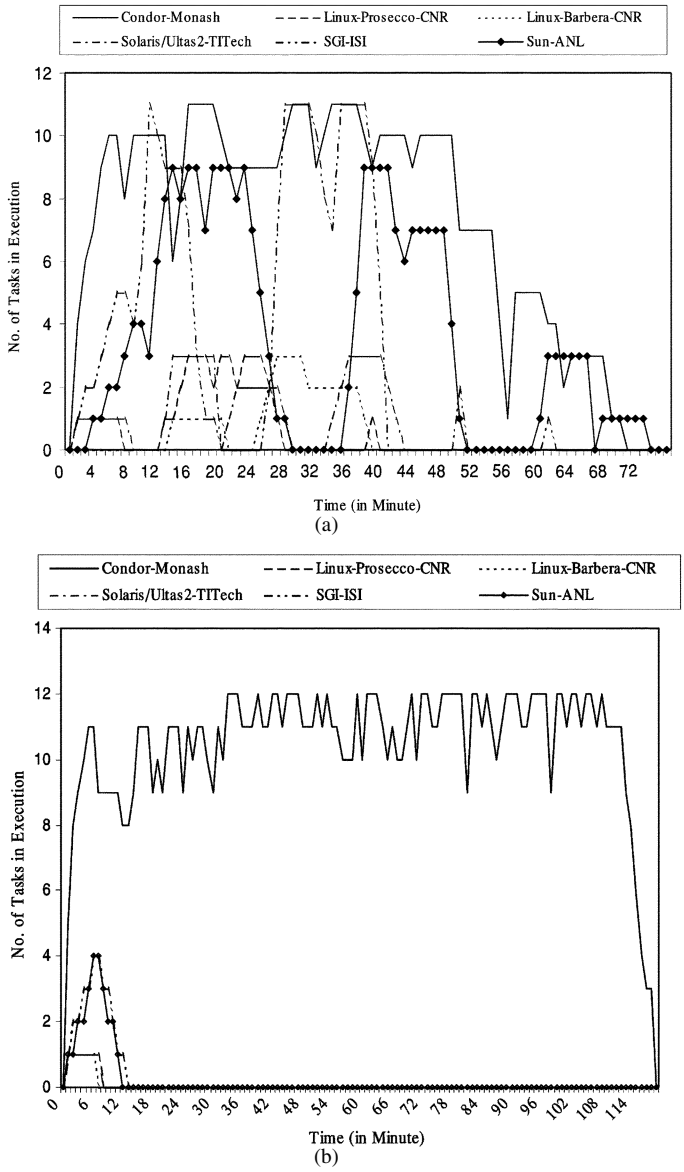


Fig. 4. (a) Resource selection in deadline and budget constrained time optimization scheduling. (b) Resource selection in deadline and budget constrained cost optimization scheduling.

time for a given budget. In this experiment, it completed execution of all jobs within 70 min and spent 237 000 G$. In the second experiment (cost minimization), the broker selected cheap resources as much as possible to minimize the execution cost whilst still trying to meet the deadline (completed in 119 min) and spent 115 200 G$. After the initial calibration phase, the jobs were distributed to the cheapest machines for the remainder of the experiment. The processing expense of the time-optimization scheduling experiment is much larger than the cost-optimization scheduling experiment due to the use of expensive resources to complete the experiment early. The results show that our Grid brokering system can take advantage of economic models and user input parameters to meet their requirements.

V. Economy Within Data Grids

A data-intensive computing environment [60] can be perceived as a real-world economic system wherein there are

Table 4 Resources Within Belle Testbed Used for Evaluation, Their Roles and Costing.

Organization	Machine details	Role	Cost (in G$ per Cpusec)	Total Jobs Executed	
				Time	Cost
Dept. of Computer Science, University of Melbourne.	*belle.cs.mu.oz.au* IBM eServer, 4 CPU, 2GB RAM, 70 GB HD, Linux	Broker host, Data host, NWS server	N.A. (Not used as a compute resource)	--	--
School of Physics, University of Melbourne	*fleagle.ph.unimelb.edu.au* PC, 1 CPU, 512 MB RAM, 70 GB HD, Linux	Replica Catalog host, Data host, Compute resource, NWS sensor	2	3	94
Dept. of Computer Science, University of Adelaide	*belle.cs.adelaide.edu.au* IBM eServer, 4 CPU (only 1 available) , 2GB RAM, 70 GB HD, Linux	Data host, NWS sensor	N.A. (Not used as a compute resource)	--	--
Australian National University, Canberra	*belle.anu.edu.au* IBM eServer, 4 CPU, 2GB RAM, 70 GB HD, Linux	Data host, Compute resource, NWS sensor	4	2	2
Dept of Physics, University of Sydney	*belle.physics.usyd.edu.au* IBM eServer, 4 CPU (only 1 available), 2GB RAM, 70 GB HD, Linux	Data host, Compute resource, NWS sensor	4	72	2
Victorian Partnership for Advanced Computing, Melbourne	*brecca-2.vpac.org* 180 node cluster (only head node used), Linux	Compute resource, NWS sensor	6	23	2

producers and consumers of data distributed geographically across multiple organizations. Producers are entities which generate the data and control its distribution via mirroring at various replica locations around the globe. They lay down policies for replication that are guided by various criteria such as minimum bandwidth, storage and computational requirements, data security and access restrictions and data locality issues. However, information about the data replicas is assumed to be available through a data catalogue mechanism such as the Globus Replica Catalog [67]. An example of such a system would be the tier-level model proposed by the MONARC [46] group within CERN for replicating the data produced by the Large Hadron Collider (LHC) [36], [71] for use within the ATLAS and CMS collaborations. The consumers in this system would be the users or, by proxy, their applications which need to analyze this data to produce meaningful results. The users may want to investigate specific datasets out of a set of hundreds and thousands and may have specific application requirements that need not be fulfilled at every computational site.

While scheduling of bandwidth or storage-intensive applications is currently a hot topic in the Grid research community [34], [38], most of them give concentrate on the job turnaround time and give less importance to the limitations of the network and storage resources such as bandwidth and storage caps or increasing cost with increasing usage. Also, in large collaborative environments such as scientific Grids, the presence of a large number of users can put a lot of pressure on the data infrastructure (i.e., network and storage elements). The pressure becomes more acute when a nontrivial percentage of the users are interested in the same datasets at the same time, thus causing heavy load on the servers on which the required datasets and its replicas are hosted. This denies service to not only the requestors of the datasets in question but also to those who require other datasets that are stored on the same servers. Such an effect is commonly observed in the Internet and the World Wide Web and a popular term, "Slashdot effect," is associated with it [59].

While a robust and adaptive replication mechanism can alleviate some of the above problems, the same problems of data access and transfer costs affect the effectiveness and efficiency of such a mechanism. Previous work in economy on data grids has tried to provide solutions to this problem through economy-driven data replication mechanisms [70]. Some cost models have been proposed for such economy-based replication [22]. However, no study has been made so far on the economic aspects of data processing by scheduling analysis jobs on various sites with varying execution, transfer, and storage costs.

In this section, we extend the notion of user-driven economic scheduling within computational grids to data grids. As in economy-based computational scheduling presented in Section IV above, the user supplies the deadline by which he wants his data analysis to be completed and his budget for the analysis job. He also supplies the minimization (cost or time) he wants to apply for scheduling the jobs. As is with the case in the sections above, these inputs are directly motivated by the priority and the urgency with which the user perceives his analysis task. Market forces guide this system to a stable state as users with less urgency will tend to save on their expenses by specifying longer deadlines.

A. Scheduling Algorithms

We have extended the previous work on economy-based scheduling and resource allocation within computational grids described above to data grids by implementing the cost and time minimization algorithms for distributed data-oriented applications. We define the cost to be minimized to be the sum of the processing cost, the data transfer (network) cost and the storage cost. Likewise, the time to be minimized is the sum of the job completion time and the data transfer time. The heuristic that we have implemented is a variation

of the Min-Min heuristic described in [42] and is detailed below:

1) Repeat for every scheduling interval while there exists unprocessed jobs
2) For every job, find the data file(s) that it is dependent on and locate the data hosts for those files.
3) Find a data-compute set (a set consisting of one compute resource for the execution and one data host for each file involved) that guarantees the *minimum cost* for that job.
4) Sort the jobs in the order of increasing cost.
5) Assign jobs from the sorted list starting with the *least expensive* job until either all the jobs are done or all the compute resources have been allocated their maximum *job limit*.

The *job limit* of each compute resource is calculated each scheduling interval on the basis of its present and past job completion ratio, i.e., the ratio of the number of jobs completed to the total number of jobs allocated but not completed since the previous scheduling interval. While the above listing shows only cost minimization, the same heuristic was followed in the case of time minimization except that the criterion in the second step was changed to the *minimum execution time required*.

B. Scheduling Experiments

To evaluate the scheduling heuristics, we have used an experimental setup modified from the one described in [68]. The testbed used in our experiments is detailed in Table 4. The Grid Service Broker [68], developed as part of the Gridbus Project [57], was extended to consider the price of transferring data over network links between the compute resources and the data hosts while scheduling jobs. In our experiments, although we have artificially assigned data transmission costs shown in Table 5, they can be linked to real costs as prescribed by ISPs (Internet Service Providers). We have used NWS (Network Weather Service) [63] for measuring the network bandwidths between the computational and the data sites. A number of the data hosts were also functioning as compute resources. The bandwidth between a data host and a compute resource on the same site was set to an arbitrarily high value (10000 Mbps) within the broker and the cost of network transfer in this case is set to zero.

We have also extended the synthetic parameter sweep application *calc* used for evaluating deadline and budget algorithms for computational grids in Section IV.A to be used on data grids. The extension implements transfer and processing of large data files that are located through querying the replica catalog as described in [68]. There are 100 data files, each 30 MB in size. Each of the five data hosts in Table 5 holds an equal number of these files and there is no replication of these files. Each job depends on exactly one of the input data files, thus creating 100 jobs. Since the output files are small (in KB) and are transferred to the broker upon completion of the jobs, we have not considered storage costs within this evaluation.

The experiments were carried out on 9th August 2004 between 6:00 p.m. and 10:00 p.m. AEST. Table 6 shows the summary of the results that were obtained. As is expected, cost minimization scheduling produces minimum computation and data transfer expenses whereas time minimization completes the experiments in the least time. The graphs in Fig. 5(a) and 5(b) show the number of jobs completed against time for the two scheduling strategies for data grids. It can be seen that these mirror the trends within the graphs in Fig. 4(a) and 4(b) for computational grids, i.e., time minimization used the more expensive but faster resources to execute jobs whereas cost minimization used the cheaper resource most to ensure a lower overall expense.

Tables 8 and 7 show the number of jobs that were allocated to a unique pair of a data host and a compute resource—the number of jobs which were executed on the compute node within the pair and accessed data from the corresponding data node—for time minimization and cost minimization, respectively. It can be seen that within time minimization, the jobs were allocated to the *best available* compute resource *nearest* (highest bandwidth available) to the source of data. For example, most of the jobs dealing with the data from ANU were executed at the University of Sydney node and those dealing with data from University of Melbourne Computer Science

Table 5 Network Costs Between the Data Hosts and the Compute Resources (in G$ Per MB)

Data Node / Compute Node	ANU	UniMelb Physics	Sydney Physics	VPAC
ANU	0	34.0	31.0	38.0
Adelaide CS	34.0	36.0	31.0	33.0
UniMelb Physics	40.0	0	32.0	39.0
UniMelb CS	36.0	30.0	33.0	37.0
Sydney Physics	35.0	33.0	0	37.0

Table 6 Summary of Evaluation Results

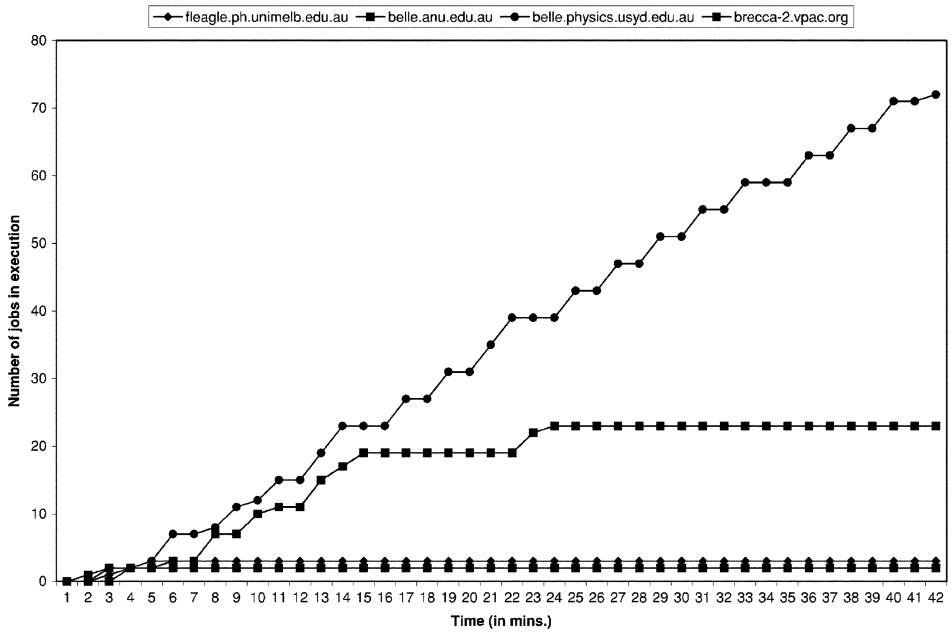
Scheduling strategy	Total Time Taken (mins.)	Compute Cost (G$)	Data Cost (G$)	Total Cost (G$)
Cost Minimization	71.07	26865	7560	34425
Time Minimization	48.5	50938	7452	58390

Table 7 Data Compute Allocation Matrix for Cost Minimization Scheduling

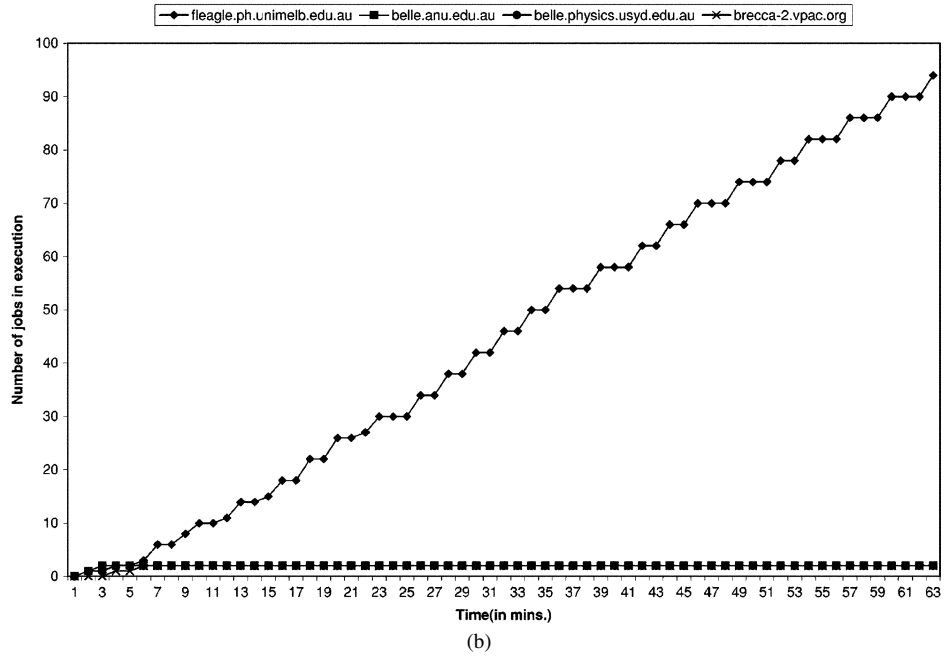
Compute Node / Data Node	UoM Phy	ANU	Sydney	VPAC
ANU	18	2	0	0
Adelaide	18	0	0	2
UoM CS	20	0	0	0
UoM Phy	20	0	0	0
Sydney	18	0	2	0

Table 8 Data Compute Allocation Matrix for Time Minimization Scheduling

Compute Node / Data Node	UoM Phy	ANU	Sydney	VPAC
ANU	0	2	18	0
Adelaide	0	0	19	1
UoM CS	0	0	2	18
UoM Phy	3	0	13	4
Sydney	0	0	20	0



(a)



(b)

Fig. 5. (a) Cumulative number of jobs completed versus time for time minimization scheduling in data grids. (b) Cumulative number of jobs completed versus time for cost minimization scheduling in data grids.

host were executed at the VPAC compute resource. Within cost minimization however, most of the jobs are allocated to the cheapest node without regard for the bandwidth cost. This is because the cost of processing dominated over the cost of transferring data, as seen from Table 6.

VI. Auction Models for Grid Resource Allocation

Auctioning has long been an important aspect of many economies. It provides a fair trading environment that has withstood the test of time. Auctions come in many different shapes and sizes, each born out of necessity or cultural differences. Despite these differences all auctions have common ideas and a basic principle that remains unchanged. Generally, every auction will consist of three entities: the seller, the bidders and the auctioneer. The auctioneer is responsible for the overall management of the auction.

In a Grid computing environment we can find entities which directly correspond to the three described in a traditional auction model. The seller is the GSP wishing to gain maximum profit out of its resources; the bidders are the Grid Resource Broker (GRB) which bid for access to resources keeping in mind the user requirements and constraints and the arbiter is Grid Market Auctioneer (GMA) which manages the auction process.

From this we can see how the auction model can be applied to a Grid computing environment. One important difference here is the correlation between assets in a traditional

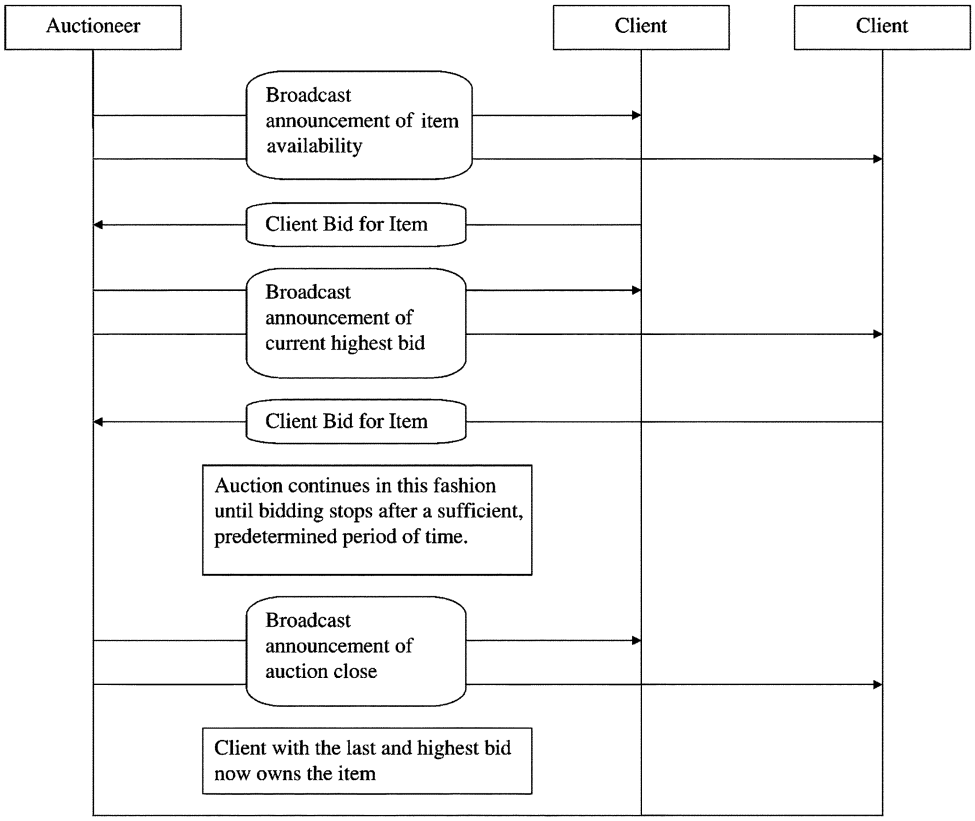


Fig. 6. The English Auction Interaction Model.

auction and resources in a Grid computing environment. In a traditional auction, the auctioneer handles all transactions whereas in a Grid resource auction, the bidders are bidding for the right to use a resource. So, once the auction is over, the auctioneer does not continue to mediate between buyer and seller. This is illustrated in the diagrams and is an important difference in how auctioning is applied to grids.

We have chosen to investigate English, Dutch and Double Auctions within Grid context.

A. English Auction

The English auction model is one of the most widely used auction models. It is an open auction where bidders can raise their bids in an attempt to outbid other prospective buyers. Fig. 6 outlines the basic interactions of the English auction.

The English auction has several positive and negative aspects. The fact that this is an open auction allows people to see what others are bidding. This helps clients to bid sensibly as well as giving you the opportunity to adjust your bid accordingly. Having many bidders is another benefit to the service providers as it allows them to sell for a much more competitive price.

One of the disadvantages of this system is the high level of communication required in this model. After each client makes a bid, the auctioneer must announce the new price to allow others to outbid. In a traditional auction this is not a problem as auctions are usually held with all bidders present, but it causes a serious problem when interested parties are on opposite sides of the world (as the case may be in a Grid computing environment). The problem of resources being priced too high is quite a common one. Judging the correct value of a temporary resource is quite difficult and relies on understanding the market comprehensively. This level of knowledge is hard to gain and would be a problem in a Grid computing environment. The last problem is that of wealthy companies outbidding smaller companies. To prevent this, brokers can be allowed to participate only in a certain number of auctions over a set time frame.

B. Dutch Auction

In the Dutch auction, depicted in Fig. 7, the auctioneer starts the price of the item high and continually lowers the price until a buyer steps in and takes the item at that price.

The Dutch auction, though conceptually similar to the English auction, has several key differences. An important advantage is that market forces of supply-and-demand play a greater role in this auction. When a service is in demand, bidders will bid earlier and hence, the service fetches a higher price. When there are excess services available against the demand, people will not bid until the price drops until it is more reasonable. The suppliers also price their services accordingly.

Though there is less communication than the English auction model, it is still at a high level. Every time the item decreases in value, an announcement is broadcast to each bidder creating a lot of communication. Pricing difficulties are not as severe as in the English model as market forces will drive prices down or up. Overvalued resources therefore, do not pose a problem. However, there is still potential for undervaluing a resource. Like the English auction, buyers in the

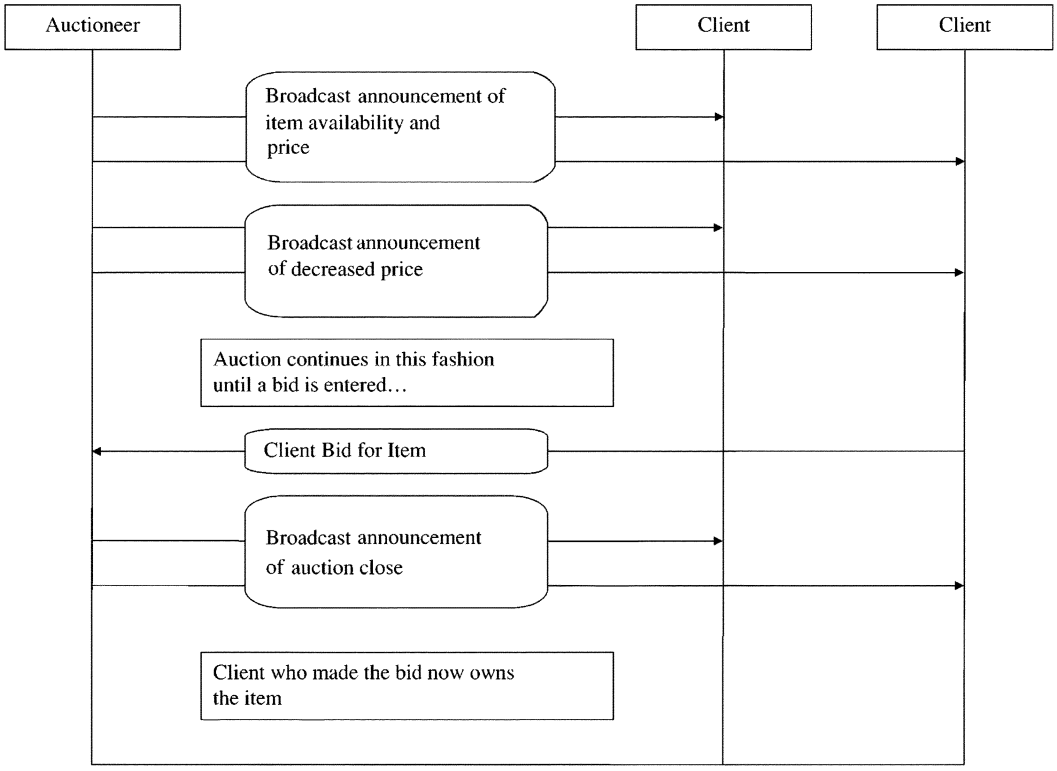


Fig. 7. The Dutch Auction Interaction Model.

Dutch auction have to judge the correct value of a service. This requires a knowledge of the market and current trends which is quite hard to come by.

C. Double Auction

The Double auction is very different from either of the previous two models. The Double auction works on a system of asks and bids. The sellers ask a price for their resources and the buyers make bids for the same. The auction therefore, is a continual process where the auctioneer matches up corresponding asks and bids to make a sale. The Double auction within a Grid context is shown in Fig. 8.

Within the Double auction, the presence of multiple simultaneous bidders for multiple sets of resources increases the throughput of the model. In a Grid computing environment, this allows one GMA to service all the GSPs and GRBs quickly and efficiently. The low levels of communication are also a positive feature. Both the seller and the bidder are only required to post the ask price and the bid price once and a message is sent back to them by the GMA announcing whether they have made a trade or not. This is much lower than in the previous models. Supply and demand are also factored into this model as trades can only be made with corresponding asks and bids. This means that what is perceived to be unreasonable will not sell.

Since this is a closed auction, it allows for private bidding but also keeps a lot of people in the dark. An unscrupulous GMA could exploit this and match similar bids and asks but keep the difference for himself. However, a system where the GMA matches bids and asks but does not carry out the transaction would be safe and this is likely to be the case in a Grid computing environment.

Table 9 Comparison Between Auction Models for Grid Environment

Auction Type	Social Welfare (Global Good)	Computational Efficieny	Communication Efficiency
English Auction	Resource is allocated to the GRB who values it the most.	Large amount of computational time spent in bidding process.	Very high levels of communication to allocate one resource. Low level of communication when demand is low.
Dutch Auction	Resource not necessarily given to the GRB who values it the most.	Less computational time required than for the English auction	Less communication per resource allocation than English auction. Higher level of communication per resource than English auction when demand is low.
Double Auction	GSPs do not have the option of taking the highest bidder during periods of high demand. Resource not necessarily given to the GRB who values it the most.	Smaller amount of computational time required per resource allocation than for the English or Dutch auction.	Smallest amount of communication per resource allocation out of the three auction models. No communication across the grid if no GRBs require resources.

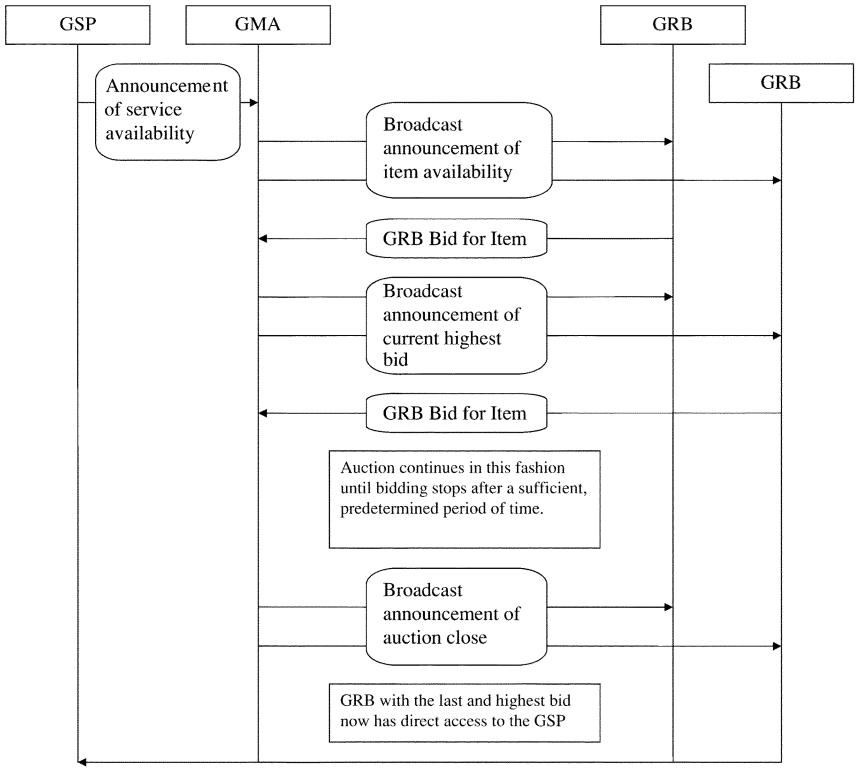


Fig. 8. The Double Auction Interaction Model.

D. Comparison and Comments

The three auction models are best compared for efficiency using some of the criteria stated in Section II. This comparison is summarized in Table 9. The Double auction uses an auction to imitate the traditional resource allocation methods. Instead of broadcasting to the whole grid when a resource becomes free for use, the GSPs only inform the GMA of the available resource. Similarly GRBs do not transmit their bids to the entire grid, just the GMA. The end result is that there is a reduction in the amount of network traffic per resource allocation and no network traffic at all when none of the GRBs require any resources.

Auctions provide a fair way of allocating resources in a grid environment despite the disadvantage of holding a resource while waiting for others. They are effective when dealing with a large number of participants and while dealing with objects whose value vary with perception.

VII. Summary and Conclusion

We have introduced computational economy as a model for tackling challenges of resource management within large-scale Grids and have discussed various approaches followed by representative works. We have proposed and discussed a reference system architecture driven by Grid economy. The use of computational economy within Nimrod-G and Gridbus brokers for compute and data intensive applications, respectively, has been presented. We have also formulated and evaluated scheduling in computational and data Grids environments. The results demonstrate effectiveness of commodity market-based resource allocation and also meet users' QoS requirements. In addition, auction models and their implications when applied to resource trading within Grid environments have been discussed.

We believe the support for economy-based resource management within Grid computing environments is essential for pushing Grids into mainstream computing. Therefore, we recommend that next-generation grids should consider economic incentive as one of the key design parameters. The importance of this has also been recognized by many national and international research agencies including the European Commission, which has identified Grid economy as one of the key thrust areas for research and development within their Next Generation Grids 2005–2010 program [19].

Acknowledgment

The authors would like to thank J. Giddy and H. Stockinger who contributed to some of the work described in this paper. R. Buyya would like to thank his Grid course students, T. Brophy and C. Talyor, who analyzed auction protocols and created the interaction diagrams based on material described in [53]. Some of the material included in this paper has been drawn from [50]–[56]. Nimrod-G has been the work of a number of people, including R. Sosic, J. Giddy, S. Garic, and C. Enticott. The Nimrod-G project has been supported by the Co-operative Research Center for Enterprise Distributed Systems (DSTC) since 1994. The contributors to the Gridbus Project include J. Yu, M. Placek, A. Barmouta, A. Sulistio, C. S. Yeo, R. Ranjan, D. C. Hong, H. Gibbins, and A. Luther. The Gridbus Project research and innovation sponsors include Australian Research Council, Storage Technology Corporation, Sun Microsystems, VPAC, IBM, and Singapore Computer Systems.

REFERENCES

[1] A. Barmouta and R. Buyya, "GridBank: A grid accounting services architecture (GASA) for distributed systems sharing and integration," presented at the Workshop Internet Computing and E-Commerce, Proc. 17th Annu. Int. Parallel and Distributed Processing Symposium (IPDPS 2003), Nice, France, Apr. 22–26, 2003.

[2] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The data grid: Toward an architecture for the distributed management and analysis of large scientific datasets," *J. Netw. Comput. Applicat.*, vol. 23, no. 3, Jul. 2000.

[3] A. Lazar and N. Semret, "Auctions for Network Resource Sharing," Columbia Univ., TR 468-97-02, Feb. 1997.

[4] A. Luther, R. Buyya, R. Ranjan, and S. Venugopal, *High Performance Computing: Paradigm and Infrastructure*, L. Yang and M. Guo, Eds, New York: Wiley. Peer-to-Peer Grid Computing and a .NET-based Alchemi Framework, Fall 2004. (in print).

[5] B. Carpenter, IPv6 and the Future of the Internet, Jul., 23 2001. The Internet Society Member Briefing.

[6] B. Chun and D. Culler, "Market-Based Proportional Resource Sharing for Clusters," University of California, Berkeley, Missouri, USA, Technical Report CSD-1092, Jan. 2000.

[7] B. Chun, "Market-based cluster resource management," Ph.D. dissertation, University of California at Berkeley, Berkeley, CA, Oct. 2001.

[8] B. Cooper and H. Garcia-Molina, "Bidding for storage space in a peer-to-peer data preservation system," in *Proc. 22nd International Conference on Distributed Computing Systems (ICDSC 2002)*, Vienna, Austria, Jul. 2–5, 2002, http://www-db.stanford.edu/peers/.

[9] C. Kenyon and G. Cheliotis, "Architecture requirements for commercializing grid resources," presented at the 11th IEEE Int. Symp. High Performance and Distributed Computing (HPDC-11), Edinburgh, U.K., Jul. 2002.

[10] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, and W. Stornetta, "Spawn: A distributed computational economy," *IEEE Trans. Softw. Eng.*, vol. 18, no. 2, pp. 103–117, Feb. 1992.

[11] D. Abramson, J. Giddy, and L. Kotler, "High performance parametric modeling with nimrod/G: Killer application for the global grid?," presented at the Int. Parallel and Distributed Processing Symposium (IPDPS 2000), Cancun, Mexico, May 1–5, 2000.

[12] D. Abramson, P. Roe, L. Kotler, and D. Mather, "ActiveSheets: Super-computing with spreadsheets," presented at the High Performance Computing Symposium (HPC'01), Advanced Simulation Technologies Conf., Seattle, WA, Apr. 2001.

[13] D. Abramson, R. Sosic, J. Giddy, and B. Hall, "Nimrod: A tool for performing parametised simulations using distributed workstations," presented at the 4th IEEE Int. Symp. High Performance Distributed Computing, Pentagon City, VA, Aug. 1995.

[14] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford, "Xenoservers: Accounted execution of untrusted code," presented at the 7th Workshop Hot Topics in Operating Systems (HotOS-VII), Rio Rico, AZ, Mar. 28–30, 1999.

[15] F. Berman and R. Wolski, "The AppLeS project: A status report," presented at the 8th NEC Research Symp., Berlin, Germany, May 1997.

[16] F. Berman, G. Fox, and T. Hey, *The Grid: Past, Present, Future, Grid Computing: Making the Global Infrastructure a Reality*. New York: Wiley, 2003.

[17] G. Fedak, C. Germain, V. Néri, and F. Cappello, "XtremWeb : A generic global computing system," presented at the 1st IEEE/ACM Int. Symp. Cluster Computing and the Grid (CCGrid 2001), Brisbane, Australia, May 15–18, 2001.

[18] G. Heiser, F. Lam, and S. Russell, "Resource management in the mungi single-address-space operating system," presented at the Australasian Computer Science Conf., Perth, Australia, Feb. 4–6, 1998.

[19] H. Bal *et al.*, "Next Generation Grid(s) European Grid Research 2005–2010," Information Society Technologies, European Commission, Expert Group Rep., Jun. 16, 2003.

[20] H. Casanova and J. Dongarra, "NetSolve: A network server for solving computational science problems," *Int. J. Supercomputing Applicat. High Performance Computing*, vol. 11, no. 3, pp. 212–223, 1997.

[21] H. Casanova, G. Obertelli, F. Berman, and R. Wolski, "The AppLeS parameter sweep template: User-level middleware for the grid," presented at the IEEE Supercomputing Conf. (SC 2000), Dallas, TX, Nov. 2000.

[22] H. Stockinger, K. Stockinger, E. Schikuta, and I. Willers, "Toward a cost model for distributed and replicated data stores," presented at the 9th Euromicro Workshop on Parallel and Distributed Processing PDP 2001, Mantova, Italy, Feb. 2001.

[23] I. Foster and C. Kesselman, Eds., *The Grid: Blueprint for a Future Computing Infrastructure*. San Mateo, CA: Morgan Kaufmann, 1999.

[24] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *Int. J. Supercomput. Applicat.*, vol. 11, no. 2, pp. 115–128, 1997.

[25] I. Foster, A. Roy, and V. Sander, "A quality of service architecture that combines resource reservation and application adaptation," presented at the IEEE/IFIP 8th Int. Workshop Quality of Service (IWQOS 2000), Pittsburgh, PA, Jun. 2000.

[26] Electricity Trading Over the Internet Begins in Six New England States (1999, May 13). [Online]. Available: http://industry.java.sun.com/javanews/stories/story2/0,1072,15093,00.html

[27] J. Almond and D. Snelling, "UNICORE: Uniform access to supercomputing as an element of electronic commerce," *Future Generation Comput. Syst.*, vol. 613, pp. 1–10, 1999.

[28] J. Bredin, D. Kotz, and D. Rus, "Utility Driven Mobile-Agent Scheduling," Dartmouth College, Hanover, NH, Tech. Rep. CS-TR98-331, Oct. 3, 1998.

[29] J. Brooke, M. Foster, S. Pickles, K. Taylor, and T. Hewitt, "Minigrids: Effective test-beds for grid application," presented at the 1st IEEE/ACM Int. Workshop Grid Computing (GRID 2000), Bangalore, India, Dec. 17, 2000.

[30] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya, "Libra: A computational economy based job scheduling system for clusters," in *Int. J. Software: Practice and Experience*, May 2004, vol. 34, pp. 573–590.

[31] J. Yu, S. Venugopal, and R. Buyya, "A Market-Oriented Grid Directory Service for Publication and Discovery of Grid Service Providers and their Services," Grid Computing and Distributed Systems (GRIDS) Laboratory, The University of Melbourne, Australia, Tech. Rep., GRIDS-TR-2003-0, Jan. 2003.

[32] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke, "A resource management architecture for metacomputing systems," presented at the 4th Int. Workshop Job Scheduling Strategies for Parallel Processing, Orlando, FL, Mar. 30, 1998.

[33] K. Czajkowski, D. Ferguson, I. Foster, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. (2004, Mar.) From Open Grid Services Infrastructure to WS-Resource Framework: Refactoring & Evolution. [Online]. Available: http://www.globus.org/wsrf/

[34] K. Ranganathan and I. Foster, "Decoupling computation and data scheduling in distributed data-intensive applications," presented at the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), Edinburgh, U.K., Jul. 2002.

[35] L. W. McKnight and J. Boroumand, "Pricing internet services: Approaches and challenges," *IEEE Comput.*, vol. 33, no. 2, pp. 128–129, Feb. 2000.

[36] Large Hadron Collider, CERN, Accessed (2004, Jan.). [Online]. Available: http://lhc-new-homepage.web.cern.ch/lhc-new-homepage/

[37] M. Baker and G. Fox, *High Performance Cluster Computing: Architectures and Systems*, R. Buyya, Ed. Upper Saddle River, NJ: Prentice-Hall PTR, 1999, vol. 1, Metacomputing: Harnessing informal supercomputers.

[38] M. Faerman, A. Su, R. Wolski, and F. Berman, "Adaptive performance prediction for distributed data-intensive applications," presented at the Supercomputing, Portland, OR, Nov. 1999.

[39] M. Frank. (2002, Apr.) *The OCEAN Project: The Open Computation Exchange & Auctioning Network* [Online]. Available: http://www.cise.ufl.edu/research/ocean/

[40] M. Huhns and L. Stephens, "Mutiagent systems and societies of agents," in *Multiagent Systems*, G. Weiss, Ed. Cambridge, MA: The MIT Press, 2000.

[41] M. Litzkow, M. Livny, and M. Mutka, "Condor—A hunter of idle workstations," presented at the 8th Int. Conf. Distributed Computing Systems (ICDCS 1988), San Jose, CA, Jan. 1988.

[42] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund, "Dynamic mapping of a class of independent tasks onto heterogeneous computing systems," *J. Parallel Distributed Computing*, vol. 59, no. 2, pp. 107–131, Nov. 1999.

[43] M. Miller and K. Drexler, "Markets and computation: Agoric open systems," in *The Ecology of Computation*, B. Huberman, Ed. Amsterdam, The Netherlands: Elsevier, 1998.

[44] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin, "An economic paradigm for query processing and data migration in mariposa," in *3rd Int. Conf. Parallel and Distributed Information Systems*, Austin, TX, Sep., 28–30 1994.

[45] (2001, Jun.). Mojo Nation. [Online]. Available: http://www.mojonation.net/

[46] (2004, Jul.) MONARC Project, CalTech. [Online]. Available: http://monarc.cacr.caltech.edu/
[47] N. Kapadia and J. Fortes, "PUNCH: An architecture for web-enabled wide-area network-computing," *Cluster Computing: J. Netw., Softw. Tools Applicat.*, vol. 2, no. 2, pp. 153–164, Sep. 1999.
[48] N. Nisan, S. London, O. Regev, and N. Camiel, "Globally distributed computation over the internet: The POPCORN project," presented at the Int. Conf. Distributed Computing Systems (ICDCS'98), Amsterdam, The Netherlands, May 26–29, 1998.
[49] R. Buyya and M. Murshed, "GridSim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," in *J. Concurrency and Computation: Practic-Experience (CCPE)*, Nov.–Dec. 2002, vol. 14, pp. 1175–1220.
[50] R. Buyya, D. Abramson, and J. Giddy, "A case for economy grid architecture for service-oriented grid computing," presented at the Int, Parallel and Distributed Processing Symposium: 10th IEEE Int. Heterogeneous Computing Workshop (HCW 2001), San Francisco, CA, Apr. 23, 2001.
[51] ——, "An economy driven resource management architecture for global computational power grids," presented at the 2000 Int. Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, NV, Jun. 26–29, 2000.
[52] ——, "Nimrod-G: An architecture for a resource management and scheduling system in a global computational grid," presented at the 4th Int. Conf. High Performance Computing in Asia-Pacific Region (HPC Asia 2000), Beijing, China, May 2000.
[53] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," *J. Concurrency Computation: Practice Experience (CCPE)*, vol. 14, no. 13–15, pp. 1507–1542, 2002.
[54] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson, "Economic models for management of resources in peer-to-peer and grid computing," presented at the Int. Conf. Commercial Applications for High-Performance Computing, Denver, CO, Aug. 20–24, 2001.
[55] R. Buyya. (1999–2004) The World-Wide Grid (WWG). [Online]. Available: http://www.buyya.com/ecogrid/wwg/
[56] ——, "Economic-based distributed resource management and scheduling for Grid computing," Ph.D. thesis, Monash Univ., Melbourne, Victoria, Australia, Apr. 12, 2002.
[57] R. Buyya and S. Venugopal, "The gridbus toolkit for service oriented grid and utility computing: An overview and status report," presented at the 1st IEEE Int. Workshop Grid Economics and Business Models (GECON 2004), Seoul, Korea, Apr. 23, 2004.
[58] R. Cocchi, S. Shanker, D. Estrin, and L. Zhang, "Pricing in computer networks: Motivation, formulation, and example," *IEEE/ACM Trans. Networking*, vol. 1, no. 6, pp. 614–627, Dec. 1993.
[59] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *Comput. Commun. Rev.*, vol. 32, no. 3, pp. 62–73, Jul. 2002.
[60] R. Moore, C. Baru, R. Marciano, A. Rajasekar, and M. Wan, "The grid: Blueprint for a new computing infrastructure," *Data Intensive Computing*, pp. 105–131, 1998.
[61] R. Smith and R. Davis, "The contract net protocol: High level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.
[62] R. Wolski, J. Plank, J. Brevik, and T. Bryan, "Analyzing market-based resource allocation strategies for the computational grid," *Int. J. High-Performance Computing Applicat.*, vol. 15, no. 3, Fall 2001.
[63] R. Wolski, N. Spring, and J. Hayes, "The network weather service: A distributed resource performance forecasting service for metacomputing," *Future Generation Computing Syst.*, vol. 15, no. 5–6, pp. 757–768.
[64] S. Chapin, J. Karpovich, and A. Grimshaw, "The legion resource management system," presented at the 5th Workshop Job Scheduling Strategies for Parallel Processing, San Juan, Puerto Rico, Apr. 16, 1999.
[65] S. Jackson, *QBank: A Resource Management Package for Parallel Computers*. Richland, WA: Pacific Northwest National Laboratory, 2000.
[66] S. Lalis and A. Karipidis, "An open market-based framework for distributed computing over the internet," presented at the 1st IEEE/ACM Int. Workshop Grid Computing (GRID 2000), Bangalore, India, Dec. 17, 2000.
[67] S. Vazhkudai, S. Tuecke, and I. Foster, "Replica selection in the globus data grid," in *1st IEEE/ACM Int. Conf. Cluster Computing and the Grid (CCGRID 2001)*, Brisbane, Australia, May 15–18, 2001.
[68] S. Venugopal, R. Buyya, and L. Winton, "A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids," Grid Computing and Distributed Systems Lab., Univ. Melbourne, Melbourne, Australia, Tech. Rep., GRIDS-TR-2004-1, Feb. 2004.
[69] T. Sandholm, "Distributed rational decision making," in *Multi-Agent Systems: A Modern Introduction to Distributed Artificial Intelligence*, G. Weiss, Ed. Cambridge, MA: The MIT Press, 2000.
[70] W. H. Bell, D. G. Cameron, R. Carvajal-Schiaffino, A. P. Millar, K. Stockinger, and F. Zini, "Evaluation of an economy-based file replication strategy for a data grid," presented at the Int. Workshop Agent-Based Cluster and Grid Computing at CCGrid, Tokyo, Japan, May, 2003.
[71] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data management in an international data grid project," presented at the 1st Int. Workshop Grid Computing (Grid 2000), Bangalore, India, 2000.
[72] Y. Amir, B. Awerbuch, and R. S. Borgstrom, "A cost-benefit framework for online management of a metacomputing system," presented at the 1st Int. Conf. Information and Computational Economy, Charleston, SC, Oct. 25–28, 1998.
[73] Y. Amir, B. Awerbuch, A. Barak A., S. Borgstrom, and A. Keren, "An opportunity cost approach for job assignment in a scalable computing cluster," in *IEEE Trans. Parallel Distributed Syst.*, Jul. 2000, vol. 11, pp. 760–768.

Rajkumar Buyya is a Senior Lecturer and the Storage Tek Fellow for Grid Computing in the Department of Computer Science and Software Engineering, University of Melbourne, Melbourne, Australia. He is also serving as the Director of the Grid Computing and Distributed Systems Laboratory. He has authored/co-authored over 100 papers and technical documents that include three books—Microprocessor $\times 86$ Programming, Mastering C++, and Design of PARAS Microkernel.

Mr. Buyya was awarded the Dharma Ratnakara Memorial Trust Gold Medal for academic excellence by Mysore and Kuvempu Universities.

David Abramson is a Professor of Computer Science in the School of Computer Science and Software Engineering (CSSE) at Monash University, Victoria, Australia. He is a project leader in the Co-operative Research Centre for Distributed Systems Nimrod Project and also Chief Investigator on two ARC funded research projects. He has published over 130 papers and technical documents. He is a co-founder of Active Tools P/L with Dr. R. Sosic, a company which was established to commercialize the Nimrod project and Guardsoft, a company focused on commercializing the Guard project. The commercial version of Nimrod is now called EnFuzion and is marketed by Axceleon Inc.

Srikumar Venugopal is working towards the Ph.D. degree in the Department of Computer Science and Software Engineering, University of Melbourne, Melbourne, Australia.

He is a Research Assistant in the Grid Computing and Distributed Systems Laboratory, University of Melbourne. He is the lead Researcher and Developer of the Gridbus Grid Service Broker and has assisted in its usage in grid-enabling several scientific applications. His research interests include grid economy, data grids, scheduling, and mobile agents.