A computational economy for grid computing and its implementation in the Nimrod-G resource broker

David Abramson [a,*], Rajkumar Buyya [b], Jonathan Giddy [c]

[a] *School of Computer Science and Software Engineering, Monash University, Caulfield Campus, Melbourne, Australia*
[b] *Grid Computing and Distributed Systems (GRIDS) Laboratory, University of Melbourne, Australia*
[c] *Department of Computer Science, Welsh e-Science Centre, Cardiff University, Cardiff, UK*

Abstract

Computational grids that couple geographically distributed resources such as PCs, workstations, clusters, and scientific instruments, have emerged as a next generation computing platform for solving large-scale problems in science, engineering, and commerce. However, application development, resource management, and scheduling in these environments continue to be a complex undertaking. In this article, we discuss our efforts in developing a resource management system for scheduling computations on resources distributed across the world with varying quality of service (QoS). Our service-oriented grid computing system called Nimrod-G manages all operations associated with remote execution including resource discovery, trading, scheduling based on economic principles and a user-defined QoS requirement. The Nimrod-G resource broker is implemented by leveraging existing technologies such as Globus, and provides new services that are essential for constructing industrial-strength grids. We present the results of experiments using the Nimrod-G resource broker for scheduling parametric computations on the World Wide Grid (WWG) resources that span five continents.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Grid computing; Computational economy; Nimrod-G broker; Grid scheduling; Resource management

1. Introduction

The accelerated development of grid computing systems has positioned them as promising next generation computing platforms. They enable the sharing, selection, and aggregation of geographically distributed resources for solving large-scale problems in science, engineering, and commerce [1,7,30]. However, application composition, resource management and scheduling in these environments is a complex undertaking. This is due to the geographic distribution of resources that are often owned by different organizations having different usage policies and cost models, and varying loads and availability patterns. To address these resource management challenges, we have proposed and developed a computational economy framework for resource allocation and regulation of supply and demand for resources. The new framework offers an incentive to resource owners for being part of the grid and motivates resource users to strike a balance between time for results delivery and heir economic cost, i.e., deadline and budget [5].

We are exploring the use of an economic paradigm for grid computing. We have developed an economy-driven grid resource broker within the Nimrod-G system that supports soft-deadline and budget-based

* Corresponding author. Tel.: +61-3-9905-1183;
fax: +61-3-9905-5146.
E-mail addresses: davida@csse.monash.edu.au (D. Abramson),
rajkumar@buyya.com (R. Buyya), j.p.giddy@cs.cf.ac.uk
(J. Giddy).

Table 1
Economics models and examples of distributed computing scheduling systems that adopted them

Economic model	Adopted by
Commodity market	Mungi [15], MOSIX [16], and Nimrod-G [1,4]
Posted price	Nimrod-G
Bargaining	Mariposa [12] and Nimrod-G
Tendering or contract-net model	Mariposa [12]
Auction model	Spawn [17] and Popcorn [18]
Bid-based proportional resource sharing	Rexec and Anemone [19]
Community, coalition, and bartering	Condor, SETI@Home [20], and MojoNation [21]
Monopoly and oligopoly	Nimrod-G broker can be used to choose between resources offered at different quality and prices

scheduling of applications on the computational grid [7]. Depending on users' quality of service (QoS) requirements, our resource broker dynamically leases grid services at runtime depending on their cost, quality, and availability. The scheduler allows minimization of time or cost within specific deadline and budget constraints.

Resource management systems need to provide mechanisms and tools that realize the goals of both service providers and consumers. The resource consumers need a *utility model*, representing their resource demand and preferences, and *brokers* that automatically generate strategies for choosing providers based on this model. Further, the brokers need to manage all issues associated with the execution of the underlying application. The service providers need *price generation schemes* that increase system utilization, as well as economic *protocols* that help them to offer competitive services. For the market to be competitive and efficient, coordination mechanisms are required that help the market reach an equilibrium price, that is, the market price at which the supply of a service equals the quantity demanded [13]. Numerous economic theories have been proposed in the literature and many commonly used economic models for selling goods and services can be employed as negotiation protocols in grid computing. Some of these market- or social-driven economic models are shown in Table 1 along with the identity of the distributed system that adopted the approach [8].

These economic models regulate the supply and demand for resources in grid-based virtual enterprises. We demonstrate the power of these models in scheduling computations using the Nimrod-G resource broker on a grid testbed, called the World Wide Grid (WWG)

spanning across five continents. Whilst it is not the goal of the system to earn revenue for the resource providers, this approach does provide an economic incentive for resource owners to share their facilities on the grid. Further, it encourages the emergence of a new service-oriented computing industry. Importantly, it provides mechanisms to trade-off QoS parameters such as deadline and computational cost, and offers incentive for users to relax their requirements. For example, a user may be prepared to accept a later deadline if the computation can be achieved more cheaply. Current grid computing toolkits and applications do not provide this functionality.

2. Nimrod-G: economics-driven grid computing environment

2.1. Objectives and goals

Nimrod-G is a tool for automated modeling and execution of parameter sweep applications (parameter studies) over global computational grids [1–3]. It provides a simple declarative parametric modeling language for expressing parametric experiments. A domain expert can easily create a plan for a parametric experiment and use the Nimrod system to submit jobs for execution. It uses novel resource management and scheduling algorithms based on economic principles. Specifically, it supports user-defined deadline and budget constraints for schedule optimizations and manages the supply and demand of resources in the grid using a set of resource trading services called Grid Architecture for Computational Economy (GRACE) [5–7].

Nimrod-G provides a persistent and programmable *task-farming engine* (TFE) that enables "plugging" *of* user-defined schedulers and customized applications or problem solving environments (e.g., ActiveSheets [23]) in place of default components. The TFE is a coordination point for processes performing resource trading, scheduling, data and executable staging, remote execution, and result collation. In the past, the major focus of our project was on creating *tools* that help domain experts to compose their legacy serial applications for parameter studies and run them on computational clusters and manually managed grids [2,3]. Our current focus is on the use of economic principles in resource management and scheduling on the grid in order to provide some measurable QoS to the end user. Real-world economic methods provide incentives for owners to contribute their resources to markets, and it also provides consumers with a basis for trading the QoS they receive against cost. That is, our focus revolves within an intersection area of grid architectures, economic principles, and scheduling optimizations (see Fig. 1), which is essential for pushing the grid into the mainstream computing.

2.2. Services and end users

Nimrod-G provides a suite of tools and services for creating parameter sweep applications, performing resource management, and scheduling applications. They are, a simple declarative programming language and associated GUI tools for creating scripts and parameterization of application input data files, and a grid resource broker with programmable entities for processing jobs on grid resources. The resource broker is made of a number of components, namely a persistent and programmable TFE, a schedule advisor, and a dispatcher, whose functionalities are discussed later. It also provides job management services that can be used for creating user-defined schedulers, steering and monitoring tools, and customized applications. Therefore, the end users that benefit from Nimrod-G tools, protocols, and services are:

- *Domain experts*: This group includes scientific, engineering, and commercial users with large-scale data-set processing requirements. Parameter applications can use Nimrod-G tools to compose them as coarse-grained data parallel, parameter sweep applications for executing on distributed resources. They can also take advantage of the Nimrod-G broker features to trade-off between a deadline and the cost of computation while scheduling application execution on the grid. This QoS aspect is important to end users, because the results are only useful if they are returned in a timely manner. Previous grid work has largely ignored this aspect of running real applications.
- *Problem solving environments developers*: Application developers can grid enable their applications with their own mechanisms to submit jobs to the Nimrod-G resource broker at runtime depending on user requirements for processing on the grid. This gives them the ability to create applications capable
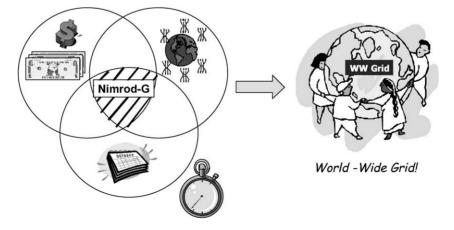

Fig. 1. QoS-based resource management—intersection of economic, scheduling, and grid worlds.

of directly using Nimrod-G tools and job management services, which in turn enables their applications for grid execution.

- *Task farming or master–worker programming environments designers*: These users can focus on designing and developing easy use and powerful application creation primitives for task farming and master–worker style programming model; developing translators and application execution environments by taking advantage of Nimrod-G runtime machinery for executing jobs on distributed grid resources.
- *Scheduling researchers*: The scheduling policy developers generally use simulation techniques and tools such as GridSim [26] and Simgrid [27] for evaluating performance of their algorithms. In simulation, it is very difficult to capture the complete property and behavior of a real-world system and hence, evaluation results may be inaccurate. Accordingly, to prove the usefulness of scheduling algorithms on actual systems, researchers need to develop runtime machinery, which is a resource intensive and time-consuming task. This can be overcome by using Nimrod-G broker programmable capability. Researchers can use Nimrod-G job management protocols and services to develop their own scheduler and associated scheduling algorithms. The new scheduler can be used to run actual applications on distributed resources and then evaluate ability of scheduling algorithms in optimally mapping jobs to resources.

2.3. Architecture

The Nimrod-G toolkit and resource broker is developed by leveraging services provided by grid middleware systems such as Globus, Legion, Condor/G, and the GRACE trading mechanisms. These middleware systems provide a set of low-level protocols for secure and uniform access to remote resources; and services for accessing resources information and storage management,. The modular and layered architecture of Nimrod-G is shown in Fig. 2. The key components of Nimrod-G resource broker consist of the following:

- Nimrod-G Clients, which can be
  - tools for creating parameter sweep applications;
  - steering and control monitors;
  - customized end user applications (e.g., ActiveSheets [23], Nimrod-O [28]).
- The Nimrod-G resource broker, that consists of
  - a TFE;
  - a scheduler that performs resource discovery, trading, and scheduling;
  - a dispatcher and actuator;
  - agents for managing the execution of jobs on resources.

The Nimrod-G broker architecture leverages services provided by lower-level different grid middleware solutions to perform resource discovery, trading, and deployment of jobs on grid resources.

2.3.1. Nimrod-G clients

2.3.1.1. Tools for creating parameter sweep applications. Nimrod supports GUI tools and declarative programming language that assist in creation of parameter sweep applications [2]. They allow user to (a) parameterize input files, (b) prepare a plan file containing the commands that define parameters and their values, (c) generate a run file, which converts the generic plan file to a detailed list of jobs and (d) control and monitor execution of the jobs. The application execution environment handles online creation of input files and command line arguments through parameter substitution.

2.3.1.2. Steering and control monitors. These components act as a user-interface for controlling and monitoring a Nimrod-G experiment. The user can vary constraints related to time and cost that influence the direction the scheduler takes while selecting resources. It serves as a monitoring console and lists the status of all jobs, which a user can view and control. A Nimrod-G monitoring and steering client snapshot is shown in Fig. 3. Another feature of the Nimrod-G client is that it is possible to run multiple instances of the same client at different locations. That means the experiment can be started on one machine, monitored on another machine by the same or different user, and the experiment can be controlled from yet another location. We have used this feature to monitor and control an experiment from Monash University and Pittsburgh Supercomputing Centre at Carnegie Melon University simultaneously during HPDC-2000 research demonstrations.
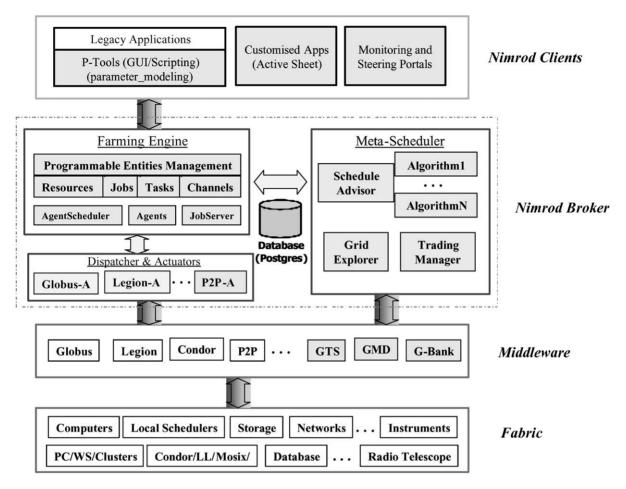
Fig. 2. Layered with an hourglass shape architecture of Nimrod-G system.

2.3.1.3. Customized end user applications. Specialized applications can be developed to create jobs at runtime and add jobs to the Nimrod-G engine for processing on the grid. These applications can use the Nimrod-G management APIs for adding and managing jobs. One such application is ActiveSheets [23], an extended Microsoft Excel spreadsheet that submits cell functions for parallel execution on computational grids using Nimrod-G services. Another example is the Nimrod-O system, a tool that uses non-linear optimization algorithms to facilitate automatic optimal design [28]. This tool has been used on a variety of case studies, including antenna design, smog modeling, durability optimization, aerofoil design and computational fluid dynamics [29].

2.3.2. The Nimrod-G grid resource broker

The Nimrod-G resource broker is responsible for determining the specific requirements that an experiment places on the grid and performing resource discovery, scheduling, dispatching jobs to remote grid nodes, starting and managing job execution, and gathering results back to the home node. The sub-modules of our resource broker are, the TFE; the scheduler that consists of a grid explorer for resource discovery, a schedule advisor backed with scheduling algorithms, and a resource trading manager; a dispatcher and actuators for deploying agents on grid resources; and agents for managing execution of Nimrod-G jobs on grid resources. The interaction between components of the Nimrod-G runtime machinery
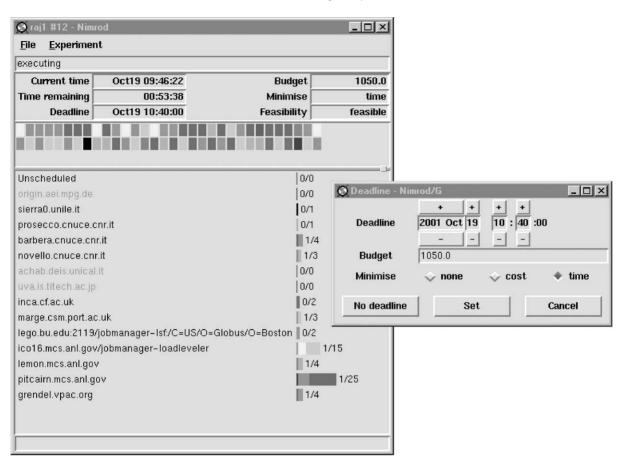
Fig. 3. A snapshot of Nimrod-G execution monitoring and steering client.

and grid services during runtime is shown in Fig. 4.

2.3.2.1. The TFE.   The Nimrod-G TFE is a persistent and programmable job control agent that manages and controls an experiment. It consists of a database to provide persistence which is accessed through a thin management interface. The farming engine is responsible for the parameterization of the experiment, the actual creation of jobs, the maintenance of job status, and provides the means for interaction between the clients, the schedule advisor, and the dispatcher. The TFE interacts with the scheduler and dispatcher in order to process jobs. It manages the experiment under the direction of schedule advisors, and then instructs the dispatcher to allocate an application task to the selected resource.

The TFE maintains the state of an entire experiment and ensures that it is recorded in persistent storage. This allows the experiment to be restarted if root node fails. The TFE exposes interfaces for job, resource, and task management along with the job-to-resource mapping APIs. Accordingly, scheduling policy developers can use these interfaces to implement other schedulers without concern for the complexity of low-level remote execution mechanisms.

2.3.2.2. The scheduler.   The scheduler is responsible for resource discovery, resource trading, resource selection, and job assignment. The resource discovery algorithm interacts with an information service (the MDS in Globus), identifies the list of authorized and available machines, trades for resource access cost, and keeps track of resource status information. The
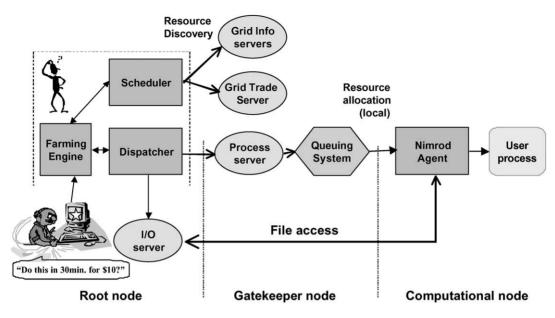
Fig. 4. The flow actions in the Nimrod-G runtime environment.

resource selection algorithm is responsible for selecting those resources that meet the deadline and budget constraints along with optimization requirements. Nimrod-G provides three different scheduling algorithms [6].

2.3.2.3. The dispatcher and actuators. The dispatcher triggers appropriate actuators to deploy agents on grid resources and assign one of the resource-mapped jobs for execution. Even though the schedule advisor creates a schedule for the entire duration based on user requirements, the dispatcher deploys jobs on resources periodically depending on load and number of CPU s that are available. We have implemented different dispatchers and actuators for each different middleware service. For example, a Globus-specific dispatcher is required for Globus resources, and a Legion-specific component for Legion resources.

2.3.2.4. Agents. Nimrod-G agents are deployed on grid resources dynamically at runtime depending on the scheduler's instructions. The agent is responsible for setting up the execution environment on a given resource for a job. It is responsible for transporting the code and data to the machine; starting the execution of the task on the assigned resource and sending

results back to the TFE. Since the agent operates on the "far side" of the middleware resource management components, it provides error detection for the user's task, sending the information back to the TFE.

The Nimrod-G agent also records the amount of resource consumed during job execution, such as the CPU time and wall clock time. The online measurement of the amount of resource consumed by the job during its execution helps the scheduler evaluate resource performance and change the schedule accordingly. Typically, there is only one type of agent for all mechanisms, irrespective of whether they are fork or queue nodes. However, different agents are required for different middleware systems.

2.4. Implementation and technologies

The Nimrod-G resource broker follows a modular, extensible, and layered architecture with an "hourglass" principle as applied in the Internet Protocol suite [11]. This architecture enables separation of different grid middleware systems *mechanisms* for accessing remote resources from the end user applications. The broker provides uniform access to diverse implementations of low-level grid services. The key components of Nimrod-G, the TFE, the

Table 2

The resource broker functionality and grid services role

Nimrod-G functionality	Grid services used or performed by
Application model	Coarse-grained task farming, master–worker, and data parallelism
Application composition	We support mechanism for application parameterization through parameterization of input files and command-line inputs for coarse-grained data parallelism. It basically supports coarse-grain, data parallel, task-farming application model, which can be expressed using our declarative programming language or GUI tools
Application interface	The Nimrod-G broker supports protocols and interfaces that help in job management. Nimrod-G clients or problem solving environments can add, remove, and inquire about job status. They can set user requirements such as deadline and budget; start and stop application execution both at job level and the entire application level
Scheduling interface	The Nimrod-G broker supports protocols and interfaces that help in developing schedulers. The schedulers can interact with the TFE, inquire jobs, user constraints, and develop a schedule that maps jobs to resources
Security	Secure access to resources and computations (identification, authentication, computational delegation) is provided by low level middleware systems like Globus
Resource discovery	Resource discovery involves discovering appropriate resources and their properties that match with users requirements. We maintain resource listings for Globus, Legion, and Condor and their static and dynamic properties are discovered using grid information services. For example, in case of Globus resources, we query Globus LDAP-based GRIS server for resource information
Resource trading and market models	The market-driven resource trading is performed using GRACE trading services. The Nimrod-G broker architecture is generic enough to support various economic models for price negotiation and using the same in developing application schedules
Performance prediction	The Nimrod-G scheduler performs the user-level resource capability measurement and load profiling by measuring and establishing the job consumption rate
Scheduling algorithms	Deadline and budget-based constrained (DBC) scheduling performed by Nimrod-G schedule advisor. Along with DBC scheduling, we support further optimization of time, cost, or surplus-driven divide and conquer in scheduling
Remote job submission	The Nimrod-G dispatcher performs deployment of Nimrod-G agents using Globus GRAM, Legion, or Condor commands. The agents are responsible for managing all aspects of job execution
Staging programs and data on remote resources	In case of Legion and Condor it is handled by their I/O management systems. On Globus resources, we use http protocols for fetching required files
Accounting (broker level)	Nimrod-G agents perform accounting tasks such as measuring resource consumptions and the scheduler performs the entire application level accounting
Monitoring and steering tools	Nimrod-G monitoring and steering client
Problem solving environments	ActiveSheets and Nimrod-O are enabled to use Nimrod-G services
Execution testbed	The WWG having resources distributed across five continents

scheduler, and the dispatcher are loosely coupled. The interaction among them takes place through network protocols. Apart from the dispatchers and the Grid Explorer, the Nimrod-G components are mechanism-independent. The modular and extensible architecture of Nimrod-G facilitates a rapid implementation of Nimrod-G support for upcoming peer-to-peer computing infrastructures such as Jxta [24] and Web services [25]. To achieve this, it is only necessary to implement two new components, a dispatcher and an enhanced Grid Explorer. The current implementation of Nimrod-G broker uses low-level

grid services provided by Globus [9] and Legion [10] systems. We also support Nimrod-G dispatcher implementation for Condor [22] resource management system. The role of various grid and commodity technologies in implementing Nimrod-G functionality and components is presented in Table 2.

While submitting applications to the broker, user requirements such as deadline and budget constraints need to be set and start application execution. These constraints can be changed at any time during execution. The complete details on application parameterization and jobs management information, starting from submission to completion, is maintained in the database. In the past the database was implemented as a file-based hierarchical database. In the latest version of Nimrod-G, the TFE database is implemented using a standard "relational" database management system.

The commodity technologies and software tools used in the Nimrod-G implementation include: the C and Python programming languages, the Perl scripting language, SQL and Embedded C for database management. The PostgreSQL database system is used for the management of the TFE database and its interaction with other components.

3. Scheduling experiments on the WWG testbed

We have performed deadline and budget constrained scheduling experiments at two different times (Australian peak and off-peak hours) on resources distributed in two major time zones [7] using a "cost-optimization scheduling algorithm" [6] on the WWG [14] testbed shown in Fig. 5. Currently, the testbed has heterogeneous computational resources owned by different organizations distributed across five continents: Asia, Australia, Europe, North America, and South America. This testbed contains heterogeneous resources such as PCs, workstations, SMPs, clusters, and vector supercomputers running operating systems such as Linux, Sun Solaris, IBM AIX, SGI IRIX and True64. Further, the systems use a variety of job management systems such as Condor, RMS, PBS, and LSF. All these resources are grid enabled using Globus services.

We have performed an experiment of 165 CPU-intensive jobs, each running approximately 5 min duration. For a given deadline of 2 h (120 min) and budget of 396,000 (G$ or tokens), we conducted experiments for two different optimization strategies:
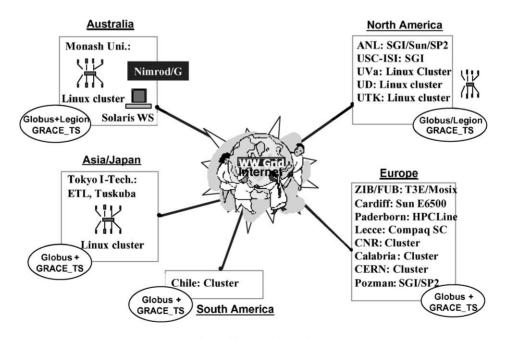

Fig. 5. The WWG testbed.

Table 3
The WWG testbed resources used in scheduling experiments, job execution and costing

Resource type and size (no. of nodes)	Organization and location	Grid services and fabric	Price (G$ per CPU s)	Jobs executed on resources	
				Time_Opt	Cost_Opt
Linux cluster (60 nodes)	Monash University, 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

1. *Optimize for time*: This strategy produces results as early as possible, but before a deadline and within a budget limit.
2. *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 [8]. It used grid resource trading services for establishing connection with the Grid Trader running on resource providers'
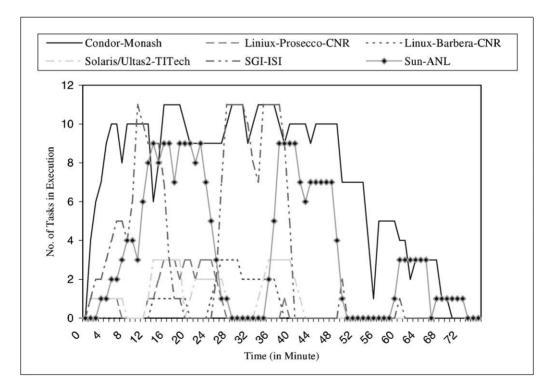

Fig. 6. Resource selection in deadline and budget-based scheduling for time optimization strategy.

machines and obtained service prices accordingly. The broker architecture is generic enough to use any of the protocols discussed in [8] for negotiating access to resources and choosing appropriate ones. The access price varies from one consumer to another and from time to time, as defined by the resource owners. 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 learn the ability of resources for executing jobs. Thus, it adapts itself to the changing resource conditions including failure of resources or jobs on the resource. The heuristics-based scheduling algorithms employed by Nimrod-G broker are presented in our early work [6].

We have used a subset of resources of the WWG testbed [14] in these scheduling experimentations. 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 GRACE resource trading protocols (commodity market model), but is based on an arbitrary assignment by us for demonstration purposes only.

The number of jobs in execution on resources (Y-axis) at different times (X-axis) during the experimentation is shown in Figs. 6 and 7 for the time and cost optimization 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 time for a given budget. In this experiment, it completed execution of all jobs within 70 min and spent G\$ 237,000. In the second experiment (cost minimization), the broker selected cheap resources as much as possible
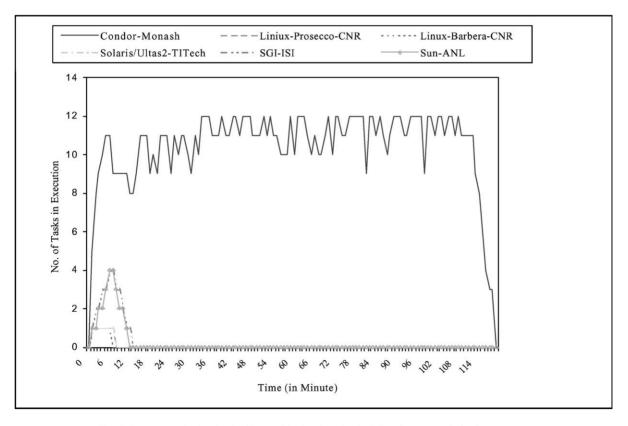

Fig. 7. Resource selection in deadline and budget-based scheduling for cost optimization strategy.

to minimize the execution cost whilst still trying to meet the deadline (completed in 119 min) and spent G\$ 115,200. After the initial *calibration phase*, the jobs were distributed to the cheapest machines for the remainder of the experiment. The cost of the time optimization experiment is much larger than cost optimization one 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.

4. Conclusions and future work

The emerging grid computing technologies are enabling the creation of virtual organizations and enterprises for sharing distributed resources for solving large-scale problems in science, engineering, and commerce. The resource management and scheduling systems in grid environments need to be adaptive to handle dynamic changes in availability of resources and user requirements. At the same time, they need to provide scalable, controllable, measurable, and easily enforceable policies for management of resources. To address these requirements, we have proposed an economic paradigm for resource management and scheduling; and developed a prototype resource broker called Nimrod-G for scheduling parameter sweep applications on distributed resources.

The Nimrod tools for modeling parametric experiments are quite mature and in production use for cluster computing. A prototype version of grid enabled tools and Nimrod-G resource broker have been implemented and they are available for download from our project Web page. The Nimrod-G TFE services have been used in developing customized clients and applications. An associated dispatcher is capable of deploying computations (jobs) on grid resources enabled by Globus, Legion, and Condor. The TFE jobs management protocols and services can be used for developing new scheduling policies. We have built a number of market-driven deadline and budget constrained scheduling algorithms, namely, time and cost optimizations with and without deadline and budget constraints. The results of scheduling experiments with different QoS requirements show

promising insights into the effectiveness of economics paradigm for management of resources and their usefulness in application scheduling with optimizations. We believe that the computational economy approach for grid computing provides one of the essential ingredients for pushing grid into the mainstream computing.

In the near future, we plan to support scheduling with advance resource reservation. The economic models that will be used are driven demand-and-supply, tenders/contract-net, and auctions protocols [8]. These new models will require suitable scheduling algorithms to be developed. All of the models and associated scheduling algorithms will be evaluated through simulations and experimentations. In order to completely automate leasing of resources at runtime and online payments, we need digital currency, which is out of the scope of our work. However, we note that electronic currency technology is rapidly progressing with emerging e-commerce infrastructure [7]; we will incorporate them in Nimrod-G when they are available.

References

[1] D. Abramson, J. Giddy, L. Kotler, High performance parametric modeling with Nimrod-G: killer application for the Global Grid, in: Proceedings of the IPDPS'2000, Mexico, IEEE CS Press, USA, 2000.

[2] D. Abramson, R. Sosic, J. Giddy, B. Hall, Nimrod: a tool for performing parameterised simulations using distributed workstations, in: Proceedings of the Fourth IEEE Symposium on High Performance Distributed Computing, Virginia, USA, August 1995.

[3] D. Abramson, I. Foster, J. Giddy, A. Lewis, R. Sosic, R. Sutherst, N. White, The Nimrod Computational Workbench: a case study in desktop metacomputing, in: Proceedings of the Australian Computer Science Conference (ACSC'97), Macquarie University, Sydney, February 1997.

[4] R. Buyya, D. Abramson, J. Giddy, Nimrod-G: an architecture for a resource management and scheduling system in a global computational grid, in: Proceedings of the HPC ASIA'2000, China, IEEE CS Press, USA, 2000.

[5] R. Buyya, D. Abramson, J. Giddy, An economy driven resource management architecture for global computational power grids, in: Proceedings of the 2000 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, USA, 26–29 June 2000.

[6] R. Buyya, J. Giddy, D. Abramson, An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications,

in: Proceedings of the Second Workshop on Active Middleware Services (AMS 2000 in conjunction with HPDC-2000), Pittsburgh, USA, 1 August 2000, Kluwer Academic Publishers, Dordrecht, 2000.

[7] R. Buyya, D. Abramson, J. Giddy, A case for economy grid architecture for service-oriented grid computing, in: Proceedings of the 10th IEEE International Heterogeneous Computing Workshop (HCW 2001 in conjunction with IPDPS 2001), San Francisco, CA, USA, April 2001.

[8] R. Buyya, H. Stockinger, J. Giddy, D. Abramson, Economic models for management of resources in peer-to-peer and grid computing, in: Proceedings of the International Conference on Commercial Applications for High-Performance Computing, 20–24 August 2001, SPIE Press, Denver, CO, USA, 2000.

[9] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke, A resource management architecture for metacomputing systems, in: Proceedings of the IPPS/SPDP'98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[10] S. Chapin, J. Karpovich, A. Grimshaw, The Legion resource management system, in: Proceedings of the Fifth Workshop on Job Scheduling Strategies for Parallel Processing, April 1999, http://legion.virginia.edu/.

[11] J. Postel, C. Sunshine, D. Cohen, The ARPA Internet Protocol, Computer Networks 5 (4) (1981).

[12] 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, in: Proceedings of the Third International Conference on Parallel and Distributed Information Systems, Austin, TX, USA, 28–30 September 1994, IEEE Computer Society Press, Los Alamitos, CA, USA, 1994.

[13] D. Ferguson, C. Nikolaou, J. Sairamesh, Y. Yemini, Economic models for allocating resources in computer systems, in: Market-Based Control: A Paradigm for Distributed Resource Allocation, World Scientific, Singapore, 1996.

[14] R. Buyya, World Wide Grid testbed, June 2001, http://www.csse.monash.edu.au/~rajkumar/ecogrid/wwg/.

[15] G. Heiser, F. Lam, S. Russell, Resource management in the Mungi single-address-space operating system, in: Proceedings of the Australasian Computer Science Conference, Perth, Australia, February 1998, Springer, Singapore, 1998.

[16] Y. Amir, B. Awerbuch, A. Barak, S. Borgstrom, A. Keren, An opportunity cost approach for job assignment in a scalable computing cluster, IEEE Trans. Parallel Distrib. Syst. 11 (7) (2000).

[17] C. Waldspurger, T. Hogg, B. Huberman, J. Kephart, W. Stornetta, Spawn: a distributed computational economy, IEEE Trans. Software Eng. 18 (2) (1992).

[18] N. Nisan, S. London, O. Regev, N. Camiel, Globally distributed computation over the Internet—the POPCORN project, in: Proceedings of the International Conference on Distributed Computing Systems (ICDCS'98), 1998.

[19] B. Chun, D. Culler, Market-based proportional resource sharing for clusters, Technical Report, University of California, Berkeley, CA, September 1999.

[20] SETI@Home, June 2001, http://setiathome.ssl.berkeley.edu/.

[21] Mojo Nation, June 2001, http://www.mojonation.net/.

[22] M. Litzkow, M. Livny, M. Mutka, Condor—a hunter of idle workstations, in: Proceedings of the Eighth International Conference of Distributed Computing Systems, June 1988.

[23] D. Abramson, P. Roe, L. Kotler, D. Mather, ActiveSheets: super-computing with spreadsheets, in: Proceedings of the 2001 High Performance Computing Symposium (HPC'01), Advanced Simulation Technologies Conference, April 2001.

[24] L. Gong, Project JXTA: a technology overview, Technical Report, Sun Microsystems Inc., April 2001, http://www.jxta.org/project/www/docs/TechOverview.pdf.

[25] S. Bansal, G. Pal, The Web at your (machine's) service, Java-World Magazine, 28 September 2001, http://www.javaworld.com/javaworld/jw-09-2001/jw-0928-smsservice.html.

[26] M. Murshed, R. Buyya, D. Abramson, GridSim: a portable and scalable toolkit for modeling and simulation of application scheduling for parallel and distributed computing, Technical Report, Monash University, October 2001, http://www.csse.monash.edu.au/~rajkumar/gridsim/.

[27] H. Casanova, Simgrid: a toolkit for the simulation of application scheduling, in: Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the grid (CCGrid 2001), Brisbane, Australia, May 2001.

[28] D. Abramson, A. Lewis, T. Peachy, Nimrod/O: a tool for automatic design optimization, in: Proceedings of the Fourth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2000), Hong Kong, China, December 2000.

[29] D. Abramson, A. Lewis, T. Peachy, C. Fletcher, An automatic design optimization tool and its application to computational fluid dynamics, in: Proceedings of the Super Computing 2001 Conference, Denver, CO, USA, November 2001.

[30] R. Buyya, K. Branson, J. Giddy, D. Abramson, The virtual laboratory: enabling molecular modeling for drug design on the World Wide Grid, Technical Report, Monash University, December 2001.

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. He 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 project: Guard—a relative debugger and a software environment for building high performance optimizing decision support systems from computational models.

Rajkumar Buyya is a lecturer in the School of Computer Science and Software Engineering at the University of Melbourne, Australia. He directs research activities of the Grid Computing and Distributed Systems (GRIDS) Laboratory. 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.

Jonathan Giddy recently joined as a coordinator for grid technologies in the Welsh e-Science Centre at Cardiff University, UK. Prior to that, he worked as 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!