Global Grid Forum:
Grid Computing Environments Community Practice (CP) Document

Project Title: Nimrod/G Problem Solving Environment and Computational Economies
CP Document Contact: Rajkumar Buyya, rajkumar@csse.monash.edu.au
Project Participants: David Abramson <davida@csse.monash.edu.au> Rajkumar Buyya <rajkumar@csse.monash.edu.au> Jonathan Giddy <jon@csse.monash.edu.au>
Project URL(s): http://www.csse.monash.edu.au/~rajkumar/ecogrid/

1. Overview

A. Description and Goals

The accelerated development in Grid and peer-to-peer computing systems has positioned them as promising next generation computing platforms. They enable the coordinated or regulated use of geographically distributed resources owned by autonomous organizations (i.e., service providers). The Grid-enabled Problem Solving Environments (PSEs) provide a secure and transparent mechanism for application composition, configuration, expression of user preferences and requirements, and automated resource recovery, scheduling, resource reservation to meet QoS requirement, management of program execution on (remote) resources including staging data and program and gathering results, online access to data sources, along with steering and status management. They leverage services provided by middleware systems for information for resource discovery, trading, advance resource reservation, secure process management, storage access, accounting, and payment management. Nimrod/G is one of the popular Grid-enabled problem solving environments, which is built by leveraging services provided by middleware systems such as Globus, Legion, GRACE, and so on [3][4].

Nimrod/G is a tool for automated modeling and execution of parameter sweep applications (parameter-studies) over global computational Grids. It provides a simple declarative parametric modeling language for expressing parametric experiments. The domain experts can easily create a plan for a parametric computing and use the Nimrod runtime system to submit legacy jobs for execution. It uses economics paradigm for resource management and scheduling on the Grid. It supports user-defined deadline and budget constraints for schedule optimisations and regulating demand and supply of resources in the Grid by leveraging services of GRACE (Grid Architecture for Computational Economy) resource trading services [5][6].

Nimrod/G provides a programmable and persistent Task *Farming Engine* that can be used for creating and plugging user-defined scheduling policies and/or customised task farming applications (e.g., ActiveSheet--that executes Microsoft Excel computations/cells on the Grid). The task farming engine coordinates resource trading, scheduling, staging data and executable, execution, and gathers results from remote Grid nodes to the user home transparently.

In the past, the major focus of our project was on creating *tools* that help domain experts (scientists, engineering, application specialists) to compose their legacy applications for parameter studies and run them on computational Clusters and manually managed Grids [1][2]. Those tools have become commodity technologies and they are even commercially available. Or current focus is on the use of economics paradigm in resource management and scheduling on the Grid. Therefore, the three dimensions of our work focus, as shown in Figure 1, are: Grid architecture, economics, and scheduling. The grid resource management and scheduling using computational economics framework provides a

well-proven real world basis for regulating the demand supply for resources and also offers incentives for resource owners to be part of Grid marketplace. It helps Grid consumers to device a strategy to tradeoff between their demand/timeframe for result delivery with computational costs. The service pricing will be driven by its value to the user at the time of experimentation and the importance of results delivery timeframe.

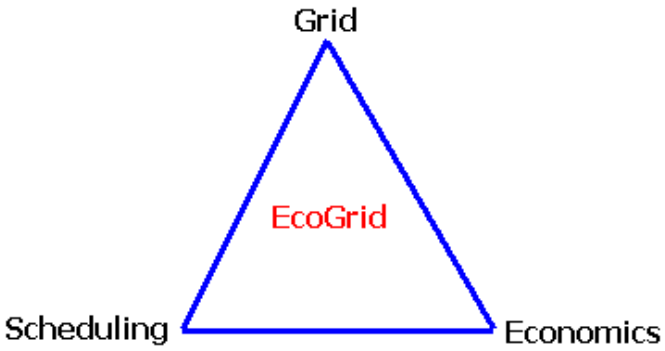

Figure 1: Market-based Resource Management and Scheduling taking Grid to its Peak!

B. Services provided

The Nimrod/G problem-solving environment provides the following key mechanisms that can be broadly grouped into Grid programming, resource management and scheduling:

- Tools/Mechanisms for Parameterising Applications (Script and GUI-based)
- Persistent, Programmable Task Farming Engine for
- Resource Discovery
- Scheduling
- Staging Data/Programs on Grid Nodes
- Steering and Execution Management
- Gathering Results

C. Systems/Sites/User Served

- Science, Engineering, and Commercial Applications that are interested parameter studies benefit from Nimrod/G
- Cutomised Applications that does online parameter studies.
- Task Decomposition and processing them in parallel (applications like complex image rendering can break large image into to segments and process each segments and render each segments simultaneously).

D. Status

- Working Prototype System is available for download.

E. Other

2. Architecture

The Nimrod/G problem-solving environment is developed by leveraging services provided by Grid milddleware systems such as Globus, Legion, Condor/G, and GRACE trading mechanisms. The middleware systems provide a set of low-level protocols for resource access and secure connectivity. A modular and layered architecture of Nimrod/G is shown in Figure 2. The key components of Nimrod/G consists of:

- Nimrod/G Clients
    - GUI-based Tools or Scripting mechanisms for Parameterising Applications
    - Steering and Control Monitors

    - Customised Applications (e.g., Active Sheets)
- Nimrod/G Resource Broker
    - Programmable and Persistent Task Farming Engine
    - Scheduler
        - Grid Explorer for Resource Discovery
        - Schedule Advisor backed with scheduling Algorithms
        - Resource Trading Manager
    - Dispatcher
    - Job-Wrappers/Agents for managing execution of Nimrod/G jobs on Grid nodes.

The Nimrod/G broker leverages services (see Table 1) provided by grid middleware systems to perform resource discovery and job execution on remote resources.

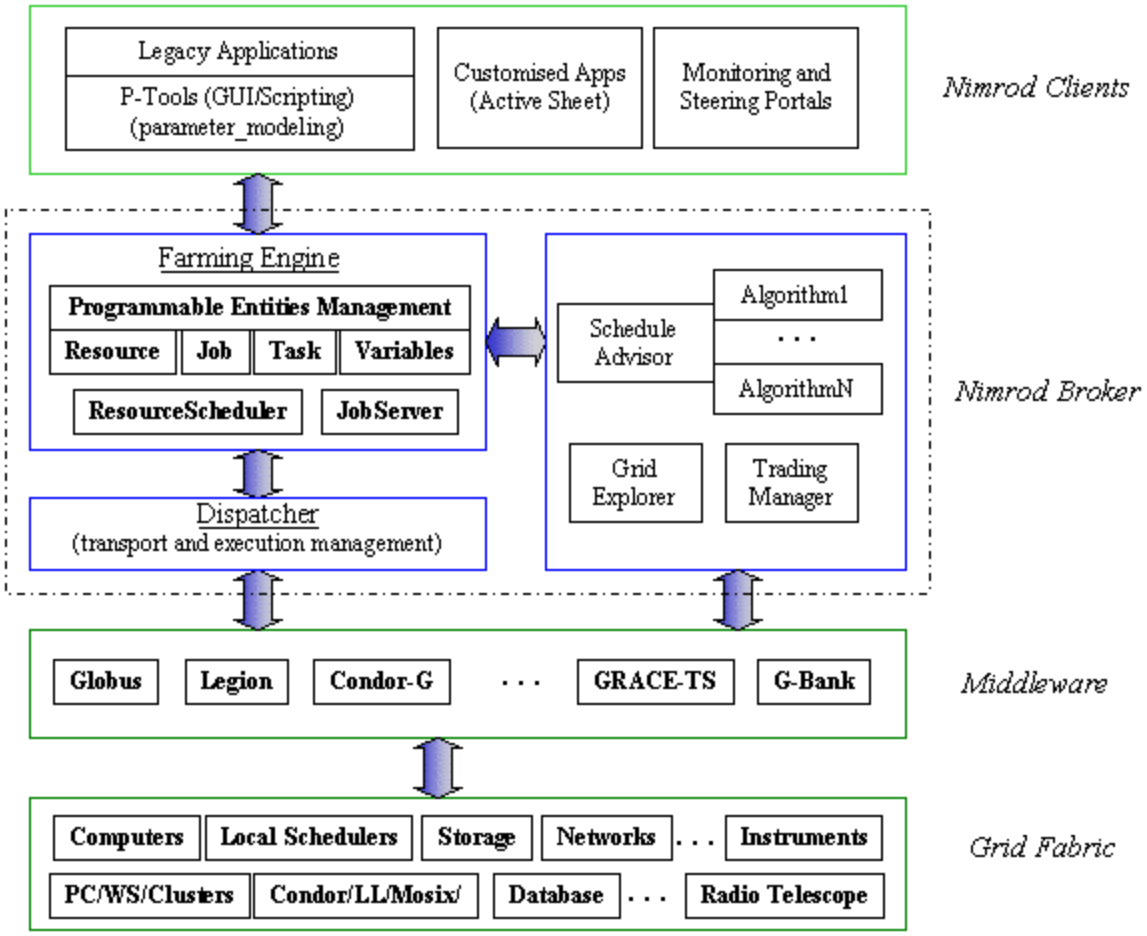

Figure 2: Nimrod/G System Architecture.

2.1 Nimrod/G Clients

Nimrod/GFarming Engine clients can be classified into:

- GUI-based Tools or Scripting mechanisms for Parameterising Applications
- Steering and Control Monitors
- Customised Applications (e.g., Active Sheets)

GUI-based Tools or Scripting mechanisms for Parameterising Applications

For rapid modeling and development of parametric applications for running on computational Grids, PSEs need to support GUI-based tools. These tools help in a) preparation of plan file that contains commands for parametrising the application inputs b) generation of run file, which converts generic plan file to detailed run_file containing list of jobs, c) converting jobs with abstract parameters to actually parameters and then adding them to the Nimrod/G engine for processing or execution. To perform first two steps, the users can use tools provided by enFuzion [8], a commercial version of Nimrod parametric modeling tool.

Steering and Control Monitors

This component acts as a user-interface for controlling and supervising an experiment under consideration. The user can vary parameters related to time and cost that influence the direction the scheduler takes while selecting resources. It also serves as a monitoring console and lists status of all jobs, which a user can view and control. 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 in Australia and Argonne National Laboratory in the USA simultaneously.

Customised Applications (e.g., Active Sheets)

Specialized applications can be developed to crate jobs at runtime and add to Nimrod/G Engine for processing on the Grid. These applications can use Nimrod/G jobs management APIs for adding and managing jobs. One such application is Active Sheets [9], an extended Microsoft Excel spreadsheet that submits cell functions for execution on computational Grids using Nimrod/G services).

2.2 Nimrod/G Grid Resource Broker

The key component of our PSE is Nimrod/G Resource broker, which is responsible determining specific demand an experiment places on the Grid and perform resource discovery, scheduling, dispatching jobs to remote Grid nodes and gathering results back to the home node along with performing management functions. The sub-modules of resource broker include,

- Programmable and Persistent Task Farming Engine
- Scheduler
  - Grid Explorer for Resource Discovery
  - Schedule Advisor backed with scheduling Algorithms
  - Resource Trading Manager
- Dispatcher
- Agents for managing execution of Nimrod/G jobs on Grid nodes.

Programmable and Persistent Task Farming Engine

The Nimrod/G Farming Engine (FE) acts as a persistent job control agent and is the central component from where the whole experiment is managed and maintained. It is responsible for parameterization of the experiment and the actual creation of jobs, maintenance of job status, interacting with clients, schedule advisor, and dispatcher. Nimrod/G clients basically add jobs to FE that works with scheduler and dispatch to complete the experimentation. It manages the experiment under the direction of schedule advisor. It then informs the dispatcher to map an application task to the selected resource.

The FE maintains the state of the whole experiment and ensures that the state is recorded in persistent storage. This allows the experiment to be restarted if the node running Nimrod goes down. The FE exposes interface for job, resource, and task management along with job-to-resource mapping APIs. These interfaces can be used by scheduling policy developers to write pluggable schedulers. Thus shielding scheduling policy developers from having to deal with a Grid's programming complexity.

Scheduler

The scheduler is responsible for resource discovery, resource selection, and job assignment. The resource discovery algorithm interacts with a grid-information service directory (the MDS in Globus), identifies the list of authorized

machines, trade for resource access cost, and keeps track of resource status information. The resource selection algorithm is responsible for selecting those resources that meet the deadline and minimize the cost of computation. We have developed three different scheduling algorithms [6].

Dispatcher

The dispatcher primarily initiates the execution of a job on the selected resource as per the scheduler's instruction. It has a abstract layer that selects an object that models job execution on Grid resource enabled by Globus/Legion/Codor-G etc. It periodically updates the status of job execution to the FE. In the current implementation, the dispatcher starts a remote component, known as Nimrod/G agent, interprets a simple script containing instructions for file transfer and execution of jobs. Based on scheduler instructions, it can even do resource reservation and assigning jobs to the reserved resource.

Nimrod/G Agent

It is basically a job-wrapper program, which is responsible for staging of application tasks and data; starting execution of the task on the assigned resource and sending results back to the FE via dispatcher. It basically mediates between FE and the actual Grid node on which the job runs.

3. Implementation/Technologies

A. Commodity technologies/software used (e.g., EJB, JMS, JINI, Perl, XML, databases...):

- Perl, Python, C, Java, Postgress with QBank and G-Bank

B. Proprietary technologies/software developed that can be shared with others

- Persistent and Programmable Task Farming Engine
- Computational Economics Protocols/Services

C. Grid Mildleware Systems Used

- Globus
- Legion
- Condor-G
- GRACE

4. Supported Grid Services

A. Define Grid software/services that the GCE currently depends upon and relationship to GF Working Group

Grid Services/Activities Leveraged	Related Global Grid Forum WG
Resource Discovery using Globus MDS (GIIS and GRIS)	Grid Information Service Working Group (GIS-WG)
Market-based Resource Trading Protocols (using GRACE trading services)	Account Management Working Group (Accounts-WG) Scheduling Working Group (Sched-WG)
Task Management using Globus GRAM	Scheduling Working Group (Sched-WG)
Resource Reservation	Scheduling and Resource Management Working Group (Sched-WG)
Staging Programs and Data on Remote Resources	Remote Data Access Working Group (Data-WG)
Secure Access to Resources and Computations (identification, authentication, computational delegation).	Security Working Group (Security-WG)
Task Farming Programming Model	Advanced Programming Models Working Group (Models-

	WG)
Measuring Resource Consumptions and Charging	Account Management Working Group (Accounts-WG)
Deadline and Budget-based Scheduling	Scheduling Working Group (Sched-WG)
Tools and Application Customizations for Desktop Access to Grid Resources	Grid Computing Environments (GCE-WG)
Resource Capability and Load Profiling (for establishing job consumption rate).	Grid Performance Working Group (Perf-WG)
Training, Supporting, and Helping Users to use Grid effectively.	User Services Working Group (Users-WG)
Customized Parametric Applications (e.g., ActiveSheets)	Application and Testbeds Working Group
InterContinental Grid Testbed	Application and Testbeds Working Group

Table 1: Grid Services leveraged by Nimrod/G and their relation to Global Grid Forum Working Groups.

B. Define Grid software/services that the GCE plans to make use of

- Advance Resource Reservation
- Market-based Resource Allocation Protocols – Contract-Net and Auctions

C. Define Grid software/services that are needed by the GCE but are not supported by the Grid

- Market-based Resource Allocation Protocols – Contract-Net and Auctions

D. Define software/services used/needed by the GCE that are outside the scope the Grid

- Electronic Currency for Exchanging Services

5. Project Status and Future Plans

Nimrod tools for modeling parametric experiments are quite mature and in production use for cluster computing. A Grid-enabled version of Nimrod (Nimrod/G) has been in development with continuous refinements and prototype version is available publically (for download over the Internet). A persistent and programmable Task Farming Engine (TFE) services has been used in developing customized clients and applications. An associated dispatcher is capable of submitting jobs to Grid resources enabled by Globus, Legion, and Condor-G. The TPE Jobs Management API can be used for creating user-defined schedulers. We have number of deadline-based Market-driven scheduling algorithms in place, including cost-optimisation with and without budget limitation and time optimization with budget limiation. In addition, all perform deadline-based scheduling.

In the near future, we will be supporting scheduling with advance resource reservation. The economic models that we will be using for resource allocation will be driven demand-and-supply, tenders/contract-net, and auctions protocols. We will be devising suitable scheduling algorithms that take advantage of these economic models. All these models and associated scheduling algorithms will be evaluated by simulations and experimentally on the Inter-Continental Grid spanning across Asia, Australia, Europe, and North America!

6. References

[1] David Abramson., Rock Sosic., Jon Giddy, and B. Hall, *Nimrod: A Tool for Performing Parametised Simulations using Distributed Workstations*, The 4th IEEE Symposium on High Performance Distributed Computing, Virginia, August 1995.

[2] David Abramson, Ian Foster, Jon Giddy, Andrew Lewis, Rock Sosic, R. Sutherst, N. White, *The Nimrod Computational Workbench: A Case Study in Desktop Metacomputing*, Australian Computer Science Conference (ACSC 97), Macquarie University, Sydney, Feb 1997.

[3] David Abramson, Jon Giddy, and Lew Kotler, *High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?*, IPDPS'2000, Mexico, IEEE CS Press, USA, 2000.

[4] Rajkumar Buyya, David Abramson, Jonathan Giddy,, *Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid*, HPC ASIA'2000, China, IEEE CS Press, USA, 2000.

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

[6] Rajkumar Buyya, Jonathan Giddy, David Abramson, *An Evaluation of Economy-based Resource Trading and Scheduling on Computational Power Grids for Parameter Sweep Applications*, The Second Workshop on Active Middleware Services (AMS 2000), August 1, 2000, Pittsburgh, USA (Kluwer Academic Press).

[7] Rajkumar Buyya, David Abramson, Jonathan Giddy, *A Case for Economy Grid Architecture for Service-Oriented Grid Computing*, 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), In conjunction with IPDPS 2001, San Francisco, California, USA, April 2001.

[8] Active Tools Inc. / TurboLinux, enFuzion Parametric Modelling Mannual: http://www.turbolinux.com/downloads/enf/man/enfuzion.htm

[9] DSTC Active Sheet project: http://www.dstc.edu.au/Research/activesheets-ov.html