Using the GridSim Toolkit for Enabling Grid Computing Education

Manzur Murshed
Gippsland School of Computing and Information Technology
Monash University, Gippsland Campus
Churchill, VIC 3842, Australia
Manzur.Murshed@infotech.monash.edu.au

Rajkumar Buyya
School of Computer Science and Software Engineering
Monash University, Caulfield Campus
Melbourne, VIC 3145, Australia
rajkumar@csse.monash.edu.au

Keywords: Grid Simulation; Education; Scheduling; Resource Management.

Abstract

Numerous research groups in universities, research labs, and industries around the world are now working on *Computational Grids* or simply *Grids* that enable aggregation of distributed resources for solving large-scale data intensive problems in science, engineering, and commerce. Several institutions and universities have started research and teaching programs on Grid computing as part of their parallel and distributed computing curriculum. The researchers and students interested in resource management and scheduling on Grid need a testbed infrastructure for implementing, testing, and evaluating their ideas. Students often do not have access to the Grid testbed and even if they have access, the testbed size is often small, which limits their ability to test ideas for scalable performance and large-scale evaluation. It is even harder to explore large-scale application and resource scenarios involving multiple users in a *repeatable* and *comparable* manner due to dynamic nature of Grid environments. To address these limitations, we propose the use of simulation techniques for performance evaluation and advocate the use of a Java-based discrete event simulation toolkit, called *GridSim*. The toolkit provides facilities for modeling and simulating Grid resources (both time and space-shared high performance computers) and network connectivity with different capabilities and configurations. We have used GridSim toolkit to simulate Nimrod-G like Grid resource broker that supports deadline and budget constrained cost and time minimization scheduling algorithms.

INTRODUCTION

The proliferation of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we do computing and use computers today. The interest in coupling of geographically distributed (computational) resources is also growing for solving large-scale problems, leading to what is popularly called *Grid* [2] and *peer-to-peer* (P2P) computing [9]. Grids enable the sharing, selection, and aggregation of suitable computational and data resources for solving large-scale data intensive problems in science, engineering, and commerce. A generic view of a Grid computing environment is shown in Figure 1. A Grid consists of four key layers of components: fabric, core middleware, user-level middleware, and applications [4]. The Grid fabric includes computers (low-end and high end computers including clusters), networks, scientific instruments, and their resource management systems. The core Grid middleware provides services that are essential for securely accessing remote resources uniformly and transparently. The services they provide include security and access management, remote job submission, storage, and resource information. The user-level middleware provides higher-level tools such as resource brokers, application development and adaptive runtime environments. Grid applications include those constructed using Grid libraries or legacy applications that can be *Grid-enabled* using user-level middleware tools.

The user essentially interacts with a *resource broker* that hides the complexities of Grid computing. The broker discovers resources that the user can access through Grid information server(s), negotiate with (Grid-enabled) resources or their agents using middleware services, map tasks to resources (*scheduling*), stages the application and data for processing (*deployment*), and finally gathers results. A resource broker is also responsible for monitoring application execution progress along with managing changes in the Grid infrastructure and resource failures. There are a number of projects worldwide actively exploring the development of various Grid computing system components, services, and applications [4].

In a Grid computing environment, the users, *producers* (also called resource owners) and *consumers*, have different goals, objectives, strategies, and demand patterns. More importantly both resources and end-users are geographically distributed with different time zones. In managing such complex Grid environments, traditional (centralized) approaches to resource management that attempt to optimize system-wide measure of performance cannot be employed. For effective utilization of distributed resources, smart resource management systems (decentralized and hierarchical or a combination [6]) are required and designing such systems is a complex undertaking. Such approaches do exist in management of resources in human economies where economic-based market models have been used in managing decentralization and heterogeneity that is present in human economies. Therefore, in [6][7][8], we proposed and explored the usage of an economics based paradigm for managing resource allocation in Grid computing environments.

The emergence of Grid as a new computing platform for parallel and distributed computing has encouraged several institutions and universities around the world to encourage research and teaching programs of Grid computing as part of their curriculum especially at graduate level courses. Some of the institutions teaching Grid computing subjects include, the University of Minnesota [10], the University of Virginia [13], the University of Southern California [14], and University of Leiden in the Netherlands [15]. A faculty teaching Grid computing at the University of Minnesota has shared his teaching experience in a recent article [11] that appeared in the IEEE Distributed Systems online magazine.

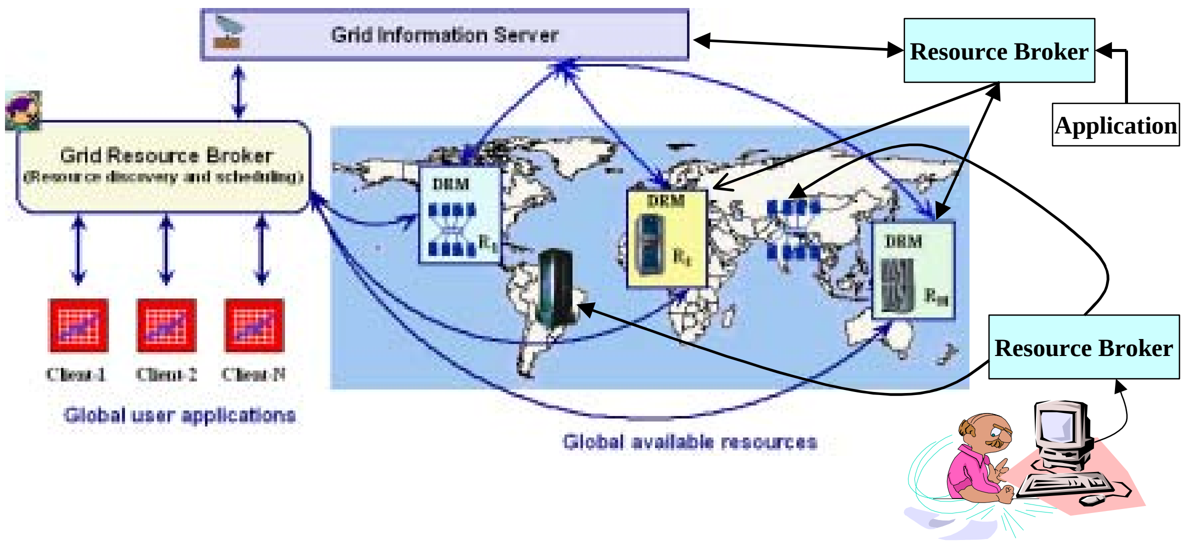

Figure 1: A generic view of a Grid computing environment.

The researchers and students investigating resource management and scheduling in Grid computing environments need a testbed infrastructure for implementing, testing, and evaluating their new or existing scheduling algorithms. For most, who do not have access to ready-to-use testbed infrastructures, building them is expensive and time consuming. Even with real testbeds, evaluating scalability and performance of scheduling mechanisms for various usage and resource scenarios is hard to achieve. It impossible to perform evaluation in a *repeatable* and *comparable* manner as the availability of testbed resources varies from time to time. To address these limitations, we propose the use of simulation techniques for performance evaluation and advocate the use of a Java-based discrete event Grid simulation toolkit, called *GridSim* [20]. It is interesting to note that the educators of cluster [12] and grid computing [11] have recognized the importance of simulation techniques and tools for modeling, simulation, and evaluation.

MOTIVATION FOR SIMULATION

The success of a peer-to-peer Grid largely depends on implementing and testing a large number of possible economic models for managing resources under various constraints [6]. It is also equally important to analyze various scheduling algorithms in obtaining certain local goals. No doubt that implementing feasible economic models and efficient scheduling algorithms in the real Grid test-bed will always remain the ultimate goal. The motivation for using simulation instead of directly using the Grid test-bed, especially in analyzing models and algorithms in the early stages, can be drawn from the following factors:

- Setting up a Grid testbed is expensive, resource intensive, and time consuming. Even if one is set up, it is mostly limited to some local area environments.
- Using a real testbed would incur real cost, as all the available resources will then be working under a true economic model. As analyzing new models and algorithms requires a large number of tests involving as many resources as available, the cost of such testing would soon become a cost burden for designing new models and algorithms. Using simulation instead of real test-bed would certainly eliminate a large portion of the above cost burden.
- Using a real testbed with real jobs is time consuming as well. Hours of real job time can be simulated in seconds provided the simulation is having sufficient processor power.
- The real test bed does not provide a *repeatable* and *controllable* environment for experimentation and evaluation of scheduling strategies.
- Simulation works well, without making the analysis mechanism unnecessary complex, by avoiding the overhead of co-ordination of real resources.
- Simulation is also effective in working with very large hypothetical problems that would otherwise require involvement of a large number of active users, which is very hard to coordinate and build at a large-scale research environment for investigation purposes.
- Simulation allows analyzing existing as well as new economic models and scheduling algorithms in the classroom.

GRID SIMULATION TOOLS

While there exist a large body of knowledge and tools: simulation languages (e.g., Parsec [16]), simulation libraries (SimJava [1]), and application specific simulators (e.g., OMNet++ network simulator [21]), there exist very few tools for simulating Grid computing environments. The notable ones are: Bricks [17], MicroGrid [19], Simgrid [18], and our GridSim [20]. The Bricks system is useful simulating client-server like global computing multi-user system, but focuses on centralized overall system performance and service rates. The MicroGrid system is a Globus emulator and expects applications and scheduler to be constructed using Globus toolkit and evaluation of large-scale Grid scenarios and configuration takes huge amount of real-time. The Simgrid supports modeling of resources that are time-shared and restricted to single user environment. It is targeted for developing schedulers that support application execution time span minimization. Unlike Simgrid, the GridSim toolkit supports both time and space-share resources along with multiuser environment. It is targeted not only for *time span minimization schedulers*, but also for those that support *deadline and budget constrained scheduling* algorithms driven by market-based economic models.

We believe that GridSim's ability to support modeling of uni- or multi-processor, shared- or distributed-memory, machines with time- or space-shared resource management appeals to wide range of scheduler designers. Its Java-based design makes it portable and available on all computational platforms. This feature appeals to educators and students since Java has emerged as one of the most popular programming language for network computing.

Salient features of our object-oriented GridSim toolkit include the following:

- It allows modeling of heterogeneous types of resources.
- Resources can be modeled operating under space- or time-shared mode.
- Resource capability can be defined (in the form of MIPS as per SPEC benchmark).
- Resources can be located in any time zone.
- Weekends and holidays can be mapped depending on resource's local time to model non-Grid (local) workload.
- Resources can be booked for advance reservation.
- Applications with different parallel application models can be simulated.
- Application tasks can be heterogeneous and they can be CPU and/or I/O intensive.
- It does not limit number of application tasks that can be submitted to a resource.
- Multiple user entities can submit tasks for execution simultaneously in the same resource, which may be time-shared or space-shared. This feature helps in building schedulers that can use different market-driven economic models for selecting services competitively.
- Network speed between resources can be specified.
- It supports simulation of both static and dynamic schedulers.
- Statistics of all or selected operations can be recorded and they can be analyzed using GridSim statistics analysis methods.

Depending on the type of resources modeled, GridSim can simulate various types of parallel and distributed computing infrastructures such as clusters, Grids, and peer-to-peer systems. In case of clusters, all the resources (called nodes) are modeled with a single time zone. Nodes can be single processor machines or *Symmetric-Multi-Processors* (SMPs), managed using time-shared operating systems, interconnected using fairly high-speed network, and resources can be put in dedicated mode. In the case of Grids, the nodes are distributed in a variety of time zones that can be single processors, SMPs, or clusters, managed by time or space-shared resource managers.

BUILDING SIMULATIONS WITH GRIDSIM

The Java-based GridSim discrete event simulation toolkit provides Java classes that represent entities essential for application, resource modeling, scheduling of jobs to resources, and their execution along with management. A schematic representation of interaction between GridSim entities during simulator execution is shown in Figure 2. The process of resource and application modeling for developing Grid schedulers/brokers is discussed below.

Resource Modeling

In GridSim toolkit, we can create CPUs (also called *Processing Elements* (PEs)) with different MIPS (Million Instructions Per Second) or SPEC-like ratings. Then, one or more PEs can be put together to create a machine (a single CPU/SMP node). Such one or more machines can be put together to create a Grid resource. The resulting Grid resource can be a single processor, shared memory multiprocessors (SMP), or a distributed memory cluster of computers. These Grid resources can be managed by time-shared or space shared scheduling systems depending on type of the resource. Generally, a single PE or SMP type Grid resource is managed by time-shared operating systems using round-robin scheduling policy and a cluster-like Grid resource is managed by space-shared Q-schedulers using different scheduling policies such as first-come-first-served (FIFO), back filling, shortest-job-first (SJF), and so on.

For every Grid resource, the non-Grid (local) workload is estimated based on typically observed load conditions as well as the time zone of the resource. The network communication speed between user and resources are defined in terms of data transfer baud rate. When a resource entity is created, it registers resource information and contact details with the Grid Information Service (GIS) entity. This resource registration process is similar to GRIS (Grid Resource Information Server) registering with GIIS (Grid Index Information Server) in Globus system. The GIS can then be queried for list of resource in a given Grid domain to get resource handles that can be used to query resources directly for their capabilities, costs, and other configurations.

Application Modeling

GridSim does not define any specific application model explicitly. It is up to the developers (of schedulers and resource brokers) to define them. We have experimented with task-farming application model and we believe that other parallel application models such as process parallelism, DAGs, divide and conquer, etc., described in [3], can also be modeled and simulated using GridSim.

In GridSim, each independent task can be heterogeneous in terms of processing time and input files size. Such tasks/jobs can be created and their requirements can be defined through *Gridlet* objects. A Gridlet is a tiny GridApp that contains all information related to jobs and job execution management details such as jobs processing requirements, expressed in MIPS, disk I/O operations, the size of input files, etc. that help in computing execution time of remote resource and the size of output files. The GridSim toolkit supports a wide range of Gridlet management protocols and services that allows one to map or assign a Gridlet to a resource for execution and manage the same through out its life cycle.

Each Grid user can be modeled with different characteristics/requirements such as:

- Types of job created e.g., job execution time, number of parametric replications etc.;
- Scheduling policy e.g., cost, time, or both minimization;
- Activity rate e.g., how often it creates new job;
- Time zone; and
- D- and B-factors, measured in the range [0,1], express deadline and budget affordability of the user.

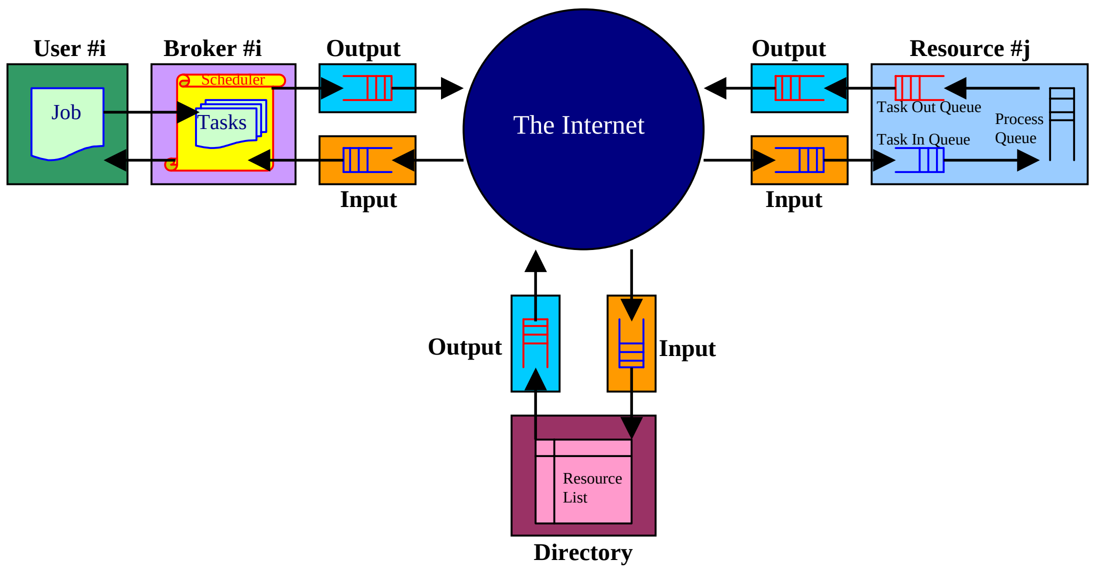

Figure 2: A flow diagram in GridSim based simulations.

D- factor close to 1 signifies that the user is willing to set deadline of a job as long as required even when only the few slowest resources are available for the job. Similarly B-factor close to 1 signifies that the user is capable of spending as much money as required even when only the few expensive resources are available for the job. These factors are basically useful for determining deadline and budget values for a given scenario at runtime. Instead, users can also explicitly specify values for deadline and budget constraints similar to the way it is currently done in Nimrod-G.

Steps for Simulating Application Scheduling

In this section we present high-level steps, with sample code clips, to demonstrate how GridSim can be used to simulate a Grid environment to analyze some application scheduling algorithms:

- First, we need to create Grid resources of different capability/speed like those in the real environment at different time zones and different policies (like time- or space-shared resources in World-Wide Grid (WWG) testbed). We also need to create users with different requirements. A sample code, serving the above purposes, is given in Figure 3.
- Second, we need to model applications by creating a number of Gridlets (that appear similar to Nimrod-G jobs) and define all parameters associated with jobs as shown in Figure 4. The Gridlets can be grouped together depending on application model for processing.
- Finally, we need to implement resource brokers as shown in Figure 5. First, inquire Grid Information Service (GIS), then inquire for resource capability including cost, and then depending on scheduling heuristics, strategy, or algorithms assign Gridlets to resources for execution. The scheduling policies can be systems-centric like those implemented in many Grid systems such as Condor-G or users-centric like Nimrod-G broker quality of services (QoS) and market-based economic models driven Deadline and Budget Constrained (DBC) time, cost, and both optimizations and policies [7].

```
public static void CreateSampleGridEnvironement(int
 no_of_users, int no_of_resources, double B_factor,
 double D_factor, int policy, double how_long, double
 seed) {
  Calendar now = Calendar.getInstance();
  GridSimController.InitSimulation(no_of_users,
   no_of_resources, now);
  // Create Resources
  for(int i=0; i<no_of_resources; i++) {
      // Create PEs
      PEList peList = new PEList();
      for(int j=0; j<(i*1+1); j++)
         peList.add(new PE(0, 100));
      // Create machine list
      MachineList mList = new MachineList();
      mList.add(new Machine(0, peList,
       ResourceCharacteristics.TIME_SHARED));
      // Create a resource containing machines
      ResourceCharacteristics resource = new
       ResourceCharacteristics("INTEL", "Linux",
       mList, ResourceCharacteristics.TIME_SHARED, 0.0,
       i*0.5+1.0);
      LinkedList Weekends = new LinkedList();
      Weekends.add(new Integer(Calendar.SATURDAY));
      Weekends.add(new Integer(Calendar.SUNDAY));
      LinkedList Holidays = new LinkedList();
                  // no holiday is set!
      // Setup resource as simulated entity with a name
      Sim_system.add(new GridResource("Resource_"+i,
       28000.0, seed, resource, 0.0, 0.0, 0.0, Weekends,
          Holidays));
  }
  // Create Users
  for(int i=0; i<no_of_users; i++)
      Sim_system.add(new UserEntity("User_"+i, 28000.0,
       how_long, seed, B_factor, D_factor, policy,
       60.0*60));
  // Start Simulation
  GridSimController.StartSimulation();
```

Figure 3: A sample code segment for creating Grid resource and user entities in GridSim.

```
Gridlet gl = new Gridlet(user_id, Gridlet_id,
 Gridlet_length, GridletFileSize, GridletOutputSize);
```

Figure 4: Gridlet method in GridSim.

Within such GridSim simulations, we can create multiple user entities with their own applications and requirements for scheduling on the same groups of resources. This can help in developing brokers like in the real world who need to share and compete for resources depending on their requirements and budget and deadline constraints. Having this ability to put brokers in competition enabled design and evaluation of market-based economic models and corresponding scheduling algorithms to create a Grid marketplace as discussed in [9].

```
class Broker extends GridSim {
  private Sim_port user;
  private Experiment experiment;
  private LinkedList ResIDList;
  private LinkedList BrokerResourceList;

  public Broker(String name, double baud_rate) {
    super(name, baud_rate);
    user = new Sim_port("user");
    add_port(user);
    GridletDispatched = 0;
    GridletReturned = 0;
    Expenses = 0.0;
    MaxGridletPerPE = 2;
  }

  …  // Gridlet management code

  public void body() {
    …  // Event handling code

    // RESOURCE DISCOVERY
    sim_schedule(output, GridSimTags.SCHEDULE_NOW,
     GridSimTags.RESOURCE_LIST, new IO_data(new
     Integer(get_id()), 0,
     GridSim.GridInformationServiceEntityId()));
    sim_get_next(ev); // Waiting for a response
    ResIDList = (LinkedList) ev.get_data();
      // extracting resources available in Grid

    … // RESOURCE TRADING and SORTING

    // SCHEDULING
    while (glFinishedList.size() <
     experiment.GetGridletList().size()) {
      if( (Sim_system.clock()>experiment.GetDeadline())
       || (Expenses > experiment.GetBudget()) )
          break;
      scheduled_count = ScheduleAdviser();
      dispatched_count = Dispatcher();
      received_count = Receiver();

      // Heurisitics for deciding hold condition
      if(dispatched<=0 && received<=0 &&
       glUnfinishedList.size()>0) {
          double deadline_left = experiment.GetDeadline()
           - Sim_system.clock();
          sim_hold(Math.max(deadline_left*0.01, 1.0));
      }
    }
  }
```

Figure 5: A sample code segment for creating a Grid resource broker in GridSim.

SIMULATION RESULTS

Using GridSim toolkit we simulated the deadline and budget constrained *cost-optimisation* scheduling algorithm [7] that attempts to complete an experiment of a large number of jobs as economically as possible within the deadline. High-level steps of this adaptive scheduling algorithm are discussed below:

1. RESOURCE DISCOVERY: Identify resources that can be used in this execution with their capability through Grid Information Service.
2. RESOURCE TRADING: Identify cost of each of the resources in terms of server cost per second and capability to be delivered per cost-unit.
3. SORT resources by increasing order of cost.
4. SCHEDULING: Repeat while there exists unprocessed jobs in application job list with a delay of scheduling event period or occurrence of an event

    [SCHEDULE ADVISOR with Policy]

    a. For each resource predict and establish job consumption rate through measure and extrapolation.
    b. For each resource based on its job consumption rate, predict and establish number of jobs a resource can consume by application deadline.
    c. For each resource in order
        i. If the number of jobs currently assigned is less than predicated number of jobs that a resource can consume, assign more jobs from unassigned job queue or from most expensive machines based on job state and feasibility.
        ii. Else if a resource has excess jobs that it can complete, take away those jobs from a resource and move to unassigned job queue.

    [DISPATCHER with Policy]

    d. The dispatcher takes care of submission of jobs to remote machines with submission and resource policy and constraints depending on resource type (time or space shared).

We simulated a Grid with independent users as many as 21 that are competing for resources as many as 25 with cost variation using normal distribution from 10 units per second (G$/sec.) to 20 G$/sec. Each user application composed of 20 jobs with variation of ±2 with random submission. Each job was modelled to 50 time units on a standard machine. The resources capability rating varied with normal distribution from 0.5 to 1.5 (with 1 for standard machine). For the sake of simplicity in analysing the result, we assumed all the users and resources stochastically similar within the respective groups with very small variances in their characteristics.

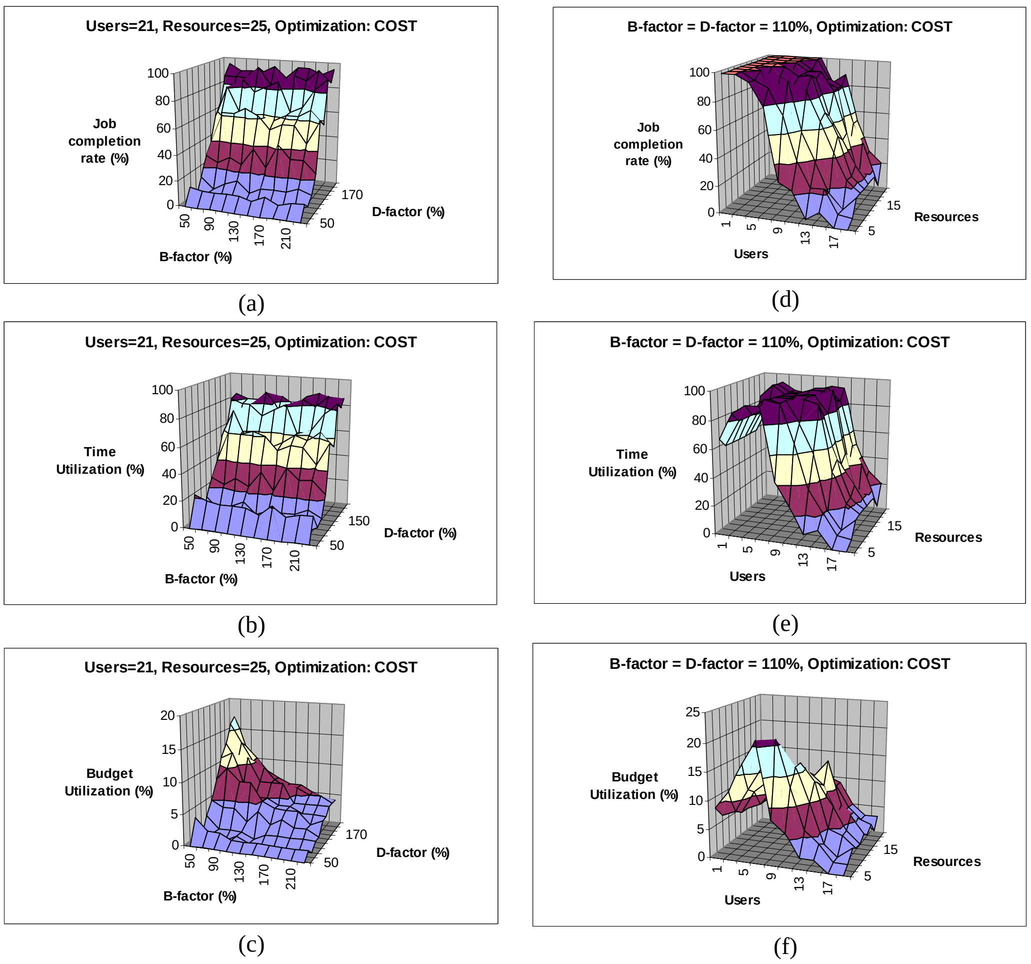

Figure 6: Job completion rate (a), time utilization (b), and budget utilization (c) of our COST_OPTIMIZATION scheduling algorithm simulated with a fixed number of users and resources while both the B- and D-factors are varied. Job completion rate (d), time utilization (e), and budget utilization (f) of the same algorithm simulated with fixed B- and D-factors while both the number of users and resources are varied.

Some of the results obtained from the experiment are plotted in Figure 6. Job completion rate, time utilization, and budget utilization are plotted in Figure 6(a)-(c) respectively, for the maximum number of users and resources used in the experiment, against the B- and D-factors that are varied from 0.5 to 2.5. These figures show that both the job completion rate and the time utilization are linearly proportional to only the D-factor that determines the deadline. On the contrary, the budget utilization varies with both the B- and D-factors. For a fixed B-factor, the budget utilization increases linearly with the D-factor; but for a fixed D-factor, decreases with the B-factor linearly, when the D-factor is low, and logarithmically, when the D-factor is reasonably high.

In Figure 6(d)-(f), we plotted job completion rate, time utilization, and budget utilization, for fixed (110%) B- and D-factors, against the number of users and resources that are varied in the ranges 1 to 21 and 1 to 25 respectively. These figures show that the job completion rate decreases with the increase in the number of users due to the obvious reason of competition. But it is also observed that both the time and budget utilizations also decrease with the increase in the number of users. This observation is against the obvious expectation that utilization of time and budget should increase with the competition. However, this unexpected result can be easily explained from the fact that both the time and budget utilizations were calculated only for successful jobs and when the job utilization is low, successful jobs, in reality, could see relatively under utilized resources.

SUMMARY AND CONCLUSION

In this paper we discussed an object oriented Grid simulation toolkit called GridSim for application scheduling simulations in Grid and P2P computing environments. The GridSim toolkit allows creation of time- and space-shared resources with different capabilities, time zones, and configurations. It supports different parallel application models that can be mapped to resources for execution by developing simulated application schedulers. We discussed architecture and components of GridSim toolkit along with steps involved in creating GridSim based application-scheduling simulators.

As we have developed Nimrod-G like Grid resource broker using GridSim and evaluated scheduling algorithms based on deadline and budget based constraints. This gave us ability to test our application scheduling policies with different Grid configurations such as varying number of resources, users, and requirements. The results are promising and therefore GridSim toolkit can be used by scheduling researchers for creating application model and perform design and evaluation of their own scheduling algorithms.

We believe that our implementation of GridSim toolkit in Java is an important move since Java-based applications are fully portable across all platforms. Java is one of the most popular programming languages for network computing and many easy-to-use and friendly development environments are available. Notably, the java virtual machine (JVM) is available for single, multiprocessor shared or distributed machines and our GridSim is multithread, which means our system is highly scalable and therefore, large-scale simulations can be performed easily. Therefore, we believe that GridSim toolkit can play an important role in enhancing Grid computing research and education.

REFERENCES

[1] F. Howell and R. McNab, *SimJava: A Discrete Event Simulation Package For Java With Applications In Computer Systems Modelling*, First International Conference on Web-based Modelling and Simulation, San Diego, CA, Society for Computer Simulation, Jan 1998.

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

[3] L. Silva and R. Buyya, *Parallel Programming Paradigms*, Chapter 2 in High Performance Cluster Computing: Programming and Applications (Vol. 2), Prentice Hall, NJ, USA, 1998.

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

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

[6] R. Buyya, D. Abramson and J. Giddy, Economy Driven Resource Management Architecture for Computational Power Grids, *Intl. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000)*, USA.

[7] R. Buyya, J. Giddy, D. 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), In conjunction with HPDC 2001, August 1, 2000, Pittsburgh, USA (Kluwer Academic Press).

[8] R. Buyya, D. Abramson, and J. Giddy, *An Economy Grid Architecture for Service-Oriented Grid Computing*, 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), with IPDPS 2001, SF, California, USA, April 2001.

[9] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson, *Economic Models for Management of Resources in Peer-to-Peer and Grid Computing*, Proceedings of International Conference on Commercial Applications for High-Performance Computing, August 20-24, 2001, Denver, Colorado, USA.

[10] J. Weissman, *CS 5980: Metacomputing*, University of Minnesota, Minneapolis, USA, Spring 2000, http://www-users.cs.umn.edu/~jon/Grids/

[11] J. Weissman, Grids in the Classroom, *IEEE Distributed Systems Online*, Volume 1, No. 3, 2000, http://www.computer.org/dsonline/archives/ds300/ds3eduprint.htm

[12] Apon, R. Buyya, H. Jin, and J. Mache, *Cluster Computing in the Classroom: Topics, Guidelines, and Experiences*, Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001), Brisbane, Australia, May 15-18, 2001.

[13] M. Humphrey, *CS 851: Grid Computing*, Department of Computer Science, University of Virginia, http://www.cs.virginia.edu/~humphrey/GridComputingClass/, 2000.

[14] A. Chervenak, *Introduction to Grid Computing*, Information Sciences Institute, University of Southern California, http://www.isi.edu/~annc/classes/grid/cs599.html, 2000.

[15] H. Bos, L. Wolters, and H. Wijshoff, *Seminarium Grid Computing*, University of Leiden, The Netherlands, 2001. http://www.liacs.nl/~herbertb/courses/grid/

[16] R. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, B. Park, H. Song, *Parsec: A Parallel Simulation Environment for Complex Systems*, Vol. 31(10), IEEE Computer, October 1998.

[17] K. Aida, A. Takefusa, H. Nakada, S. Matsuoka, S. Sekiguchi, and U. Nagashima, *Performance evaluation model for scheduling in a global computing system*, The International Journal of High Performance Computing Applications, Vol. 14, No. 3, 2000.

[18] H. Casanova, *SimGrid: A Toolkit for the Simulation of Application Scheduling*, Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001), May 15-18, 2001, Brisbane, Australia.

[19] H. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura and A. Chien, *The MicroGrid: a Scientific Tool for Modeling Computational Grids*, Proceedings of IEEE Supercomputing (SC 2000), Nov. 2000, Dallas, USA.

[20] M. Murshed, R. Buyya, D. Abramson, *GridSim: A Toolkit for the Modeling and Simulation of Global Grids*, Technical Report, Monash-CSSE 2001/102, Monash University, Australia, November 2001.

[21] Varga, *The OMNeT++ Discrete Event Simulation System*, Proceedings of the European Simulation Multiconference (ESM'2001). June 6-9, 2001. Prague, Czech Republic.